AD-A064 780 HARVARD UNIV CAMBRIDGE MA DIV OF APPLIED SCIENCES F/G 5/1
RESOURCE MANAGEMENT IN LARGE SYSTEMS.(U)
DEC 78 R SURI N00014-75-C-0648
UNCLASSIFIED TR-671 NL

1 OF 3
AD
A064780

LEVEL II

(11)

Office of Naval Research

Contract N00014-75-C-0648 NR-372-012

National Science Foundation Grant ENG-76-11824

RESOURCE MANAGEMENT IN LARGE SYSTEMS

By

Rajan Suri

December 1978

Technical Report No. 671

Division of Applied Sciences

Harvard University Cambridge, Massachusetts

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) TR- Technical Report 671	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) RESOURCE MANAGEMENT IN LARGE SYSTEMS (9)	5. TYPE OF REPORT & PERIOD COVERED Doctoral Thesis Interim / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Rajan Suri	8. CONTRACT OR GRANT NUMBER(s) (15) N00014-75-C-0648, and NSF Grant ENG-76-11824
9. PERFORMING ORGANIZATION NAME AND ADDRESS Division of Applied Sciences Harvard University Cambridge, Massachusetts 02138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS (11)	12. REPORT DATE December 1978 / 13. NUMBER OF PAGES 207
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 208 p.	15. SECURITY CLASS. (of this report) Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited. Reproduction in whole or in part is permitted by the U.S. Government.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

79 02 14 050

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Decentralization	Duality Theory
Resource Allocation	Automated Warehousing
Large-Scale Systems	File Allocation
Lagrange Multipliers	Distributed Computer Systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This work studies the application of decentralization to the problem of Resource Management in Large Systems. In an operational system, where a very large number of activities share limited resources, this Resource Management problem has three objectives. The first ("Initial Allocation") is to find an assignment of resources to every activity, such that all the system constraints are satisfied, and all activities are operating. The second ("New-Assignment") is to find a rationale for allocating resources to new activities. New activities are initiated frequently enough that we do not wish to re-solve the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

419457

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract continued

entire problem for the combined set of old and new activities. The third objective ("Periodic Review") is to find an efficient way of re-allocating resources in order to reflect the changing needs of the individual activities, as well as the changes in total resource usages.

Conventionally, the resource-allocation problem has been studied for the case where, in addition to the constraints, there exists an objective to be maximized. Our emphasis, as is reflected by the title of this work, is on the feasibility aspect of the problem, that is, of taking a large system and keeping it operational. We show that this in itself is both an important problem, and has theoretically interesting consequences. In addition, our results can be useful for the solution of the general (optimization) problem.

Chapter 2 provides the motivation for our study. The Chapter is divided into two Parts. Part I describes the Resource Management problem in a large FIAT spare-parts warehouse. We discuss why the warehouse management is satisfied with a feasible solution, and do not require any notion of optimality. The size of the problem (there are 60,000 part-numbers, and 30-50 new part-numbers are to be allocated every day), and the fact that the resource usage functions are discontinuous and/or nonlinear and/or nonconvex, make the problem a hard one. We discuss why standard techniques are not suitable. Part II of Chapter 2 advocates a decentralized approach to the problem. We show how, by replacing the feasibility problem by a suitable "Artificial" optimization problem, we can use Lagrange Multipliers to simplify the solution through decentralization of decisions. We then show how this decentralized approach offers substantial advantages over standard techniques, with regard to each of the three objectives above.

Although the use of Lagrange Multipliers for decomposition of large problems is well known, the main drawback of this technique is the existence of "duality gaps". In Chapter 3 we present a Theorem giving simple conditions which guarantee the existence of optimal multipliers. Such results have previously been given only under strict conditions. We are able to completely remove restrictions on the form of the individual resource usage functions, replacing them instead by a condition based on their magnitudes alone. This result gives a firm theoretical basis to our approach, since it justifies the use of our technique.

In Chapter 4 we develop iteration algorithms to solve the "Initial Allocation" problem in a decentralized manner. These algorithms have proveable convergence properties, and quadratic convergence rates. Again, although decentralized iteration algorithms have been described in the literature, their convergence has required strict conditions on the functions (convexity, continuity). We are able to extend our proofs to more general functions. A minor but interesting contribution of this Chapter is the Selection Algorithm. This is a simple and intuitively appealing method of solving certain inequality problems, yet its solution possesses useful minimum-norm properties.

In Chapter 5 we describe the design of a practical Resource Management system, and give examples from the system designed for the FIAT warehouse. Chapter 6 illustrates the applicability of our methods to the problem of optimal file allocation in distributed computer systems.

Our conclusions, presented in Chapter 7, advocate a new approach to Large-Scale systems. Based on the methods used in this work, we suggest that analysis of such systems should depend on the "global" properties of the systems, and be insensitive to the "local" properties of the system.

Office of Naval Research

Contract N00014-75-C-0648 NR-372-012

National Science Foundation Grant ENG 76-11824

RESOURCE MANAGEMENT IN LARGE SYSTEMS

By

Rajan Suri

Technical Report No. 671

December 1978

This report is based on the unaltered Ph.D. Thesis of Rajan Suri. The research reported in this document was made possible through support extend the Division of Applied Sciences, Harvard University by the U.S. Office of Naval Research under the Joint Services Electronics Program by Contract N00014-75-C-0648 and by the National Science Foundation under Grant ENG 76-11824.

Division of Applied Sciences

Harvard University · Cambridge, Massachusetts

Resource Management in Large Systems

A thesis presented by

Rajan Suri

to

The Division of Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

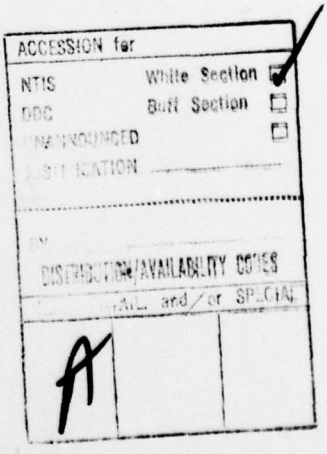
in the subject of

Engineering

Harvard University

Cambridge, Massachusetts

July 1978



Acknowledgements

During the time span over which this work has evolved, the author has received support from many individuals and institutions.

It was through my thesis Supervisor, Prof. Yu-Chi Ho, that I obtained the opportunity for investigation of this topic. I wish to express my thanks for his constant support, critical evaluation and invaluable suggestions, and especially his encouragement during moments of doubt, without which this work would never have taken shape.

I would also like to thank the other members of my committee, Prof. M. Athans of the Massachusetts Institute of Technology, and Prof. U.O. Gagliardi of Harvard University.

Different parts of this study were made possible by support extended the Division of Applied Sciences, Harvard University, by the following institutions: The Charles Stark Draper Lab., FIAT Ricambi, the U.S. Office of Naval Research under the Joint Services Electronics Program Contract N00014-75-C-0648, and the National Science Foundation under Grant ENG76-11824.

Several personnel at C.S.Draper Lab. contributed to this study. Messrs. K. Glick and S. Brosio provided constant support and encouragement. Our approach to the problem stems from a suggestion by J.S. Rhodes, who also made several contributions throughout this study. The

credit for the widespread appreciation of our results in Italy must surely go to I. Johnson for his impressive analysis of the outputs of the SALA program. During the author's time in Italy, the constant "home base" support provided by R. Asare is appreciated.

The cooperation and interest of several personnel from FIAT Ricambi made this study concrete. In particular, it was the personal encouragement of Ing. Motta that led to the development of the practical resource management system at FIAT/Volvera.

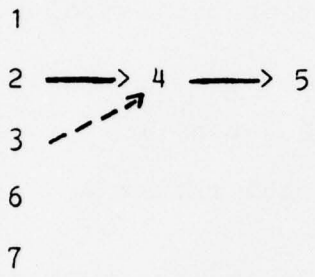
I am grateful to Mr. G. Karady of Harvard University for suggesting the investigation of the Existence Theorem in Chapter 3, and for his critical evaluations which resolved several issues therein. For helping maintain a stimulating atmosphere for creative thinking, I express fondest appreciation to Ms. Catherine Treece. In the final stages of this work, the assistance provided by Marie Cedrone is appreciated.

This thesis is dedicated to my parents and to Mala.

TABLE OF CONTENTS

Logical Dependence of Chapters

This study extends over several areas. For the benefit of readers interested in one particular subject, certain Chapters can be read independently of the others. The logical dependence of the Chapters is shown below.



KEY: A ——> B = A must be read before B

A ----> B = Results from A are used in B, but it is not essential to read A before B.

NOTATION

We state some conventions here, which will eliminate the need for minor definitions throughout this work. Underlined lower-case letters represent column vectors. Upper-case letters may represent matrices, sets, or constants. Subscripts on a symbol usually denote a component of the corresponding vector or matrix that the symbol represents, for example

x_j is the j^{th} component of the vector $\underline{x}$

A_{ij} is the (i,j) component of the matrix A.

Superscripts will be used to differentiate between symbols of the same type, for example $\underline{x}^1$, $\underline{x}^2$, $\underline{x}^k$.

Vector inequalities are to be interpreted componentwise, that is

$\underline{a} \leq \underline{b}$ means $a_i \leq b_i$ for all i.

The zero vector will often be denoted simply by 0, and its dimension will be apparent from the context.

The notation $\underline{x}'$ denotes the transpose of the vector $\underline{x}$.

E^n denotes the n-dimensional Euclidean vector space.

Synopsis

This work studies the application of decentralization to the problem of Resource Management in Large Systems. In an operational system, where a very large number of activities share limited resources, this Resource Management problem has three objectives. The first ("Initial Allocation") is to find an assignment of resources to every activity, such that all the system constraints are satisfied, and all activities are operating. The second ("New-Assignment") is to find a rationale for allocating resources to new activities. New activities are initiated frequently enough that we do not wish to re-solve the entire problem for the combined set of old and new activities. The third objective ("Periodic Review") is to find an efficient way of re-allocating resources in order to reflect the changing needs of the individual activities, as well as the changes in total resource usages.

Conventionally, the resource-allocation problem has been studied for the case where, in addition to the constraints, there exists an objective to be maximized. Our emphasis, as is reflected by the title of this work, is on the feasibility aspect of the problem, that is, of taking a large system and keeping it operational. We show that this in itself is both an important problem, and has theoretically interesting consequences. In addition, our results can be useful for the solution of the general

(optimization) problem.

Chapter 2 provides the motivation for our study. The Chapter is divided into two Parts. Part I describes the Resource Management problem in a large FIAT spare-parts warehouse. We discuss why the warehouse management is satisfied with a feasible solution, and do not require any notion of optimality. The size of the problem (there are 60,000 part-numbers, and 30-50 new part-numbers are to be allocated every day), and the fact that the resource usage functions are discontinuous and/or nonlinear and/or nonconvex, make the problem a hard one. We discuss why standard techniques are not suitable. Part II of Chapter 2 advocates a decentralized approach to the problem. We show how, by replacing the feasibility problem by a suitable "Artificial" optimization problem, we can use Lagrange Multipliers to simplify the solution through decentralization of decisions. We then show how this decentralized approach offers substantial advantages over standard techniques, with regard to each of the three objectives above.

Although the use of Lagrange Multipliers for decomposition of large problems is well known, the main drawback of this technique is the existence of "duality gaps". In Chapter 3 we present a Theorem giving <u>simple conditions</u> which <u>guarantee the existence</u> of optimal multipliers. Such results have previously been given only

under strict conditions. We are able to completely remove restrictions on the form of the individual resource usage functions, replacing them instead by a condition based on their magnitudes alone. This result gives a firm theoretical basis to our approach, since it justifies the use of our technique.

In Chapter 4 we develop iteration algorithms to solve the "Initial Allocation" problem in a decentralized manner. These algorithms have proveable convergence properties, and quadratic convergence rates. Again, although decentralized iteration algorithms have been described in the literature, their convergence has required strict conditions on the functions (convexity, continuity). We are able to extend our proofs to more general functions. A minor but interesting contribution of this Chapter is the Selection Algorithm. This is a simple and intuitively appealing method of solving certain inequality problems, yet its solution possesses useful minimum-norm properties.

In Chapter 5 we describe the design of a practical Resource Management system, and give examples from the system designed for the FIAT warehouse. Chapter 6 illustrates the applicability of our methods to the problem of optimal file allocation in distributed computer systems.

Our conclusions, presented in Chapter 7, advocate a new approach to Large-Scale systems. Based on the methods used in this work, we suggest that analysis of such systems

should depend on the "global" properties of the systems, and be insensitive to the "local" properties of the systems.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In an era where, due to rapid advances in technology, we are seeing greater and greater interconnection between systems, the study of large-scale systems is assuming a new importance. Along with this has come the realization that in most applications practicality calls for decentralized control of such systems. The importance of these two facts is reflected by the recent devotion of an entire issue of a prominent journal* to the topic of "Large-Scale Systems and Decentralized Control". In this work we study the application of decentralization to one aspect of such systems, namely, the problem of Resource Management in Large Systems.

In a large operational system, where a very large number of activities share a number of limited resources, this Resource Management problem has three main objectives. The first (the "Initial Allocation" or "Design" problem) is

---

*IEEE Trans.Aut.Control, Vol AC-23, No.2, April 1978.

to find an assignment of resources to every activity, such that all the system constraints are satisfied, and all activities are operating, enabling the system as a whole to operate. The second (the "New-Assignment" problem) is to find a rationale for allocating resources to new activities. It is presumed that new activities are initiated frequently enough that we do not wish to re-solve the entire problem for the combined set of old and new activities. The third objective ("Periodic Review" problem) is to find an efficient way of re-allocating resources in order to reflect the changing needs of the individual activities, as well as the changes in total resource usages.

Conventionally, the resource-allocation problem has been studied for the case where, in addition to the constraints, there exists an objective to be maximized. Our emphasis, as is reflected by the title of this work, is on the feasibility aspect of the problem, that is, of taking a large system and keeping it operational (maintaining it in the feasible region). We shall see that this in itself is both an important problem, and has theoretically interesting consequences. In addition, our results can be useful for the solution of the general (optimization) problem.

In Chapter 2 we shall provide the motivation for our study. The Chapter is divided into two Parts. Part I describes the Resource Management problem in a large FIAT spare-parts warehouse. It was the tackling of this problem

that led to the work presented here. Hence an appreciation of the original problem leads to a deeper understanding of the abilities of our solution technique. The resource management problem in the warehouse is exactly the general one described above, with the three objectives corresponding to (i) Initial Allocation of all parts, (ii) Allocation of new parts, and (iii) Periodic Reallocation of parts. We discuss in detail why the warehouse management is satisfied with a feasible solution, and do not require any notion of optimality. The size of the problem (there are 60,000 part-numbers, and 30-50 new part-numbers are to be allocated every day), and the fact that the resource usage functions are discontinuous and/or nonlinear and/or nonconvex, make the problem a hard one. We discuss why standard techniques (Linear Programming, Integer Programming) are not suitable.

Part II of Chapter 2 advocates a decentralized approach to the problem. We show how, by replacing the feasibility problem by a suitable "Artificial" optimization problem, we can use Lagrange Multipliers to simplify the solution through decentralization of decisions. We then show how this decentralized approach offers substantial advantages over standard techniques, with regard to each of the three objectives above. This provides us with the basic motivation for further investigation of this approach.

Although the use of Lagrange Multipliers for decomposition of large problems is well known, the main drawback of this technique is the existence of "duality gaps". This means there may not exist multipliers which can generate the optimum to the Artificial Problem above. In Chapter 3 we present a Theorem giving simple conditions which guarantee the existence of optimal multipliers. Such results have previously been given only under strict conditions (e.g. convexity, continuity). We are able to completely remove restrictions on the form of the individual resource usage functions, replacing them instead by a condition based on their magnitudes alone. This result gives a firm theoretical basis to our approach, since it justifies the use of our technique in many cases where the resource usage functions are ill-behaved and the domains of the decision variables are nonconvex.

In Chapter 4 we develop iteration algorithms to solve the "Initial Allocation" problem in a decentralized manner. These algorithms have proveable convergence properties, and quadratic convergence rates. Again, although decentralized iteration algorithms have been described in the literature, their convergence has required strict conditions on the functions (convexity, continuity). We are able to extend our proofs to more general functions. A minor but interesting contribution of this Chapter is the Selection Algorithm. This is a simple and intuitively appealing method of solving certain inequality problems, yet its

solution possesses useful minimum-norm properties.

In Chapter 5 we describe the design of a practical Resource Management system, and give examples from the system designed for the FIAT warehouse. We show how, in an operational system, our approach enables simple and efficient solution of both the "New-assignment" problem and the "Periodic Review" problem. In addition, we suggest how the algorithms in Chapter 4 can be enhanced by incorporating various features useful in a practical application.

Chapter 6 illustrates the applicability of our methods to the problem of optimal file allocation in distributed computer systems. This problem has only been addressed in the literature for a small number of files (typically 5 to 20). We show how, by appropriate formulation of the problem, a decentralized solution is possible. This brings previously intractable problems (with several thousand files, say) within the reach of known solution methods. Our aim is to demonstrate the applicability and advantages of decentralized techniques -- for the general optimization problem we do not go into details of solution algorithms, but give a framework for future research. We do however consider one case in detail -- this is the problem faced by a "Network Manager", whose task is to keep a given network operational (that is, all resource usages within the constraints) in the face of constant arrivals of new files and changing characteristics of old files.

Finally, in Chapter 7, we present our concluding remarks.

CHAPTER 2

MOTIVATION FOR THIS RESEARCH

Part I: Resource Management in a Large Warehouse

2.1 INTRODUCTION TO PART I

In this Part we motivate the reader by describing the resource management problem in a large warehouse. It was the tackling of this problem that led to the work presented in this study. Hence an appreciation of the original problem leads to a deeper understanding of the abilities and advantages of our approach.

Ofcourse, the application of our work is not restricted to the specific problem in this particular warehouse. The reader will see that our models and methods can be applied to a variety of large systems which face similar problems. In Chapter 6, for instance, we will discuss how our approach could be applied to the File Allocation Problem in large distributed computer systems.

2.2 DESCRIPTION OF WAREHOUSE

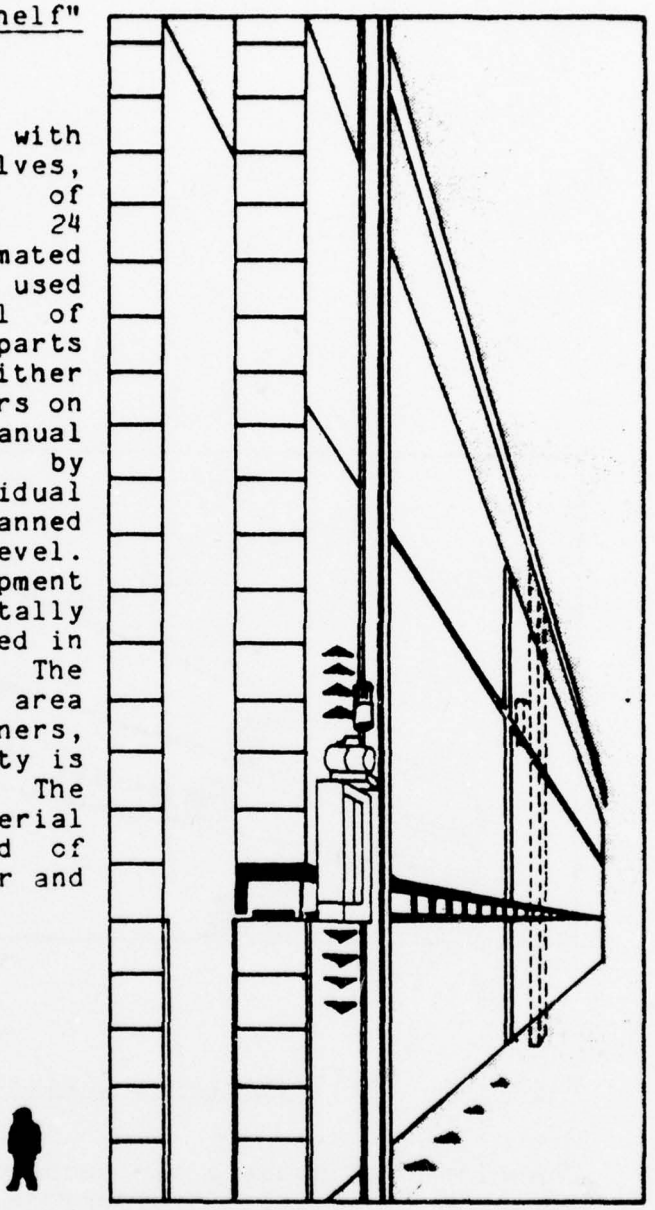
Our interest in this problem arises from a project involving the author, along with a team from CSDL*, to improve the operation of the FIAT Central Spare Parts Warehouse, in Volvera (Turin, Italy). This Warehouse essentially supplies spare parts to the whole world. It covers an area exceeding that of 15 football fields, has an inventory of over 20,000 tons (valued at several hundred million dollars), contains more than 60,000 different Part-Numbers (each of which may occupy several containers), and services about 10,000 orders every day [F1].

The Warehouse is divided into several different areas, used for stocking Parts with different characteristics. For instance, medium-sized items with not too high demand are stocked in a "High Shelf Area", where loading and retrieval of containers is done solely by computer-controlled cranes. On the other hand, very small, fast-moving items are stored in an area where they are hand-picked by men with hand-pushed carts. Figs.2-1 to 2-3 illustrate the characteristics of three such areas.

---

*The Charles Stark Draper Laboratory, Cambridge, Mass.

Fig.2-1: The "High Shelf" Area



This area is equipped with very high (27 metre) shelves, occupying a floor area of 144x96 metres, with 24 corridors served by automated cranes. The cranes are used for loading and retrieval of containers. Picking of parts from this area is done either by placing the containers on conveyors, leading to manual picking stations, or by retrieving individual containers down to manned "bays" at the floor level. Operation of all the equipment in the plant is totally automatic and is controlled in real time by a computer. The storage capacity of this area is over 120,000 containers, and the picking capacity is over 5,000 picks per day. The area is designed for material of average dimension, and of medium or high turnover and withdrawal frequency.
(Source: [F1])

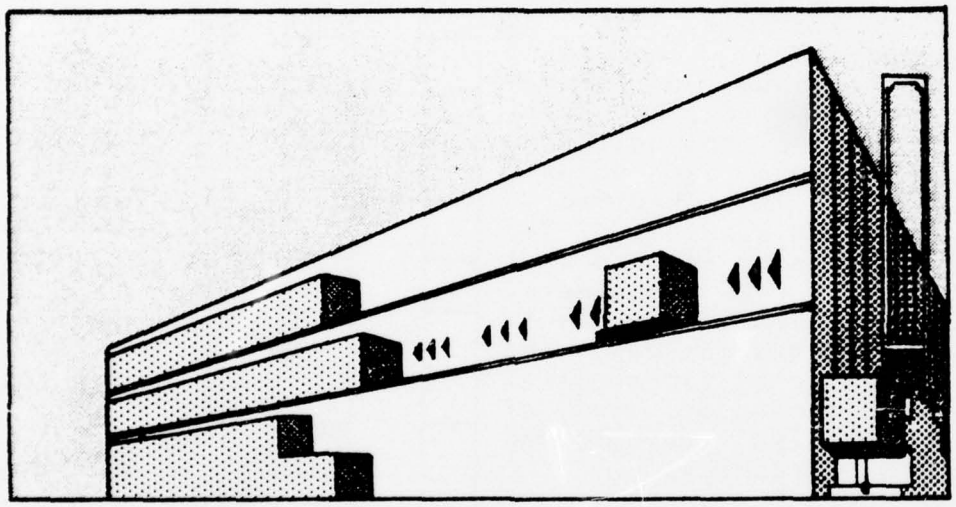
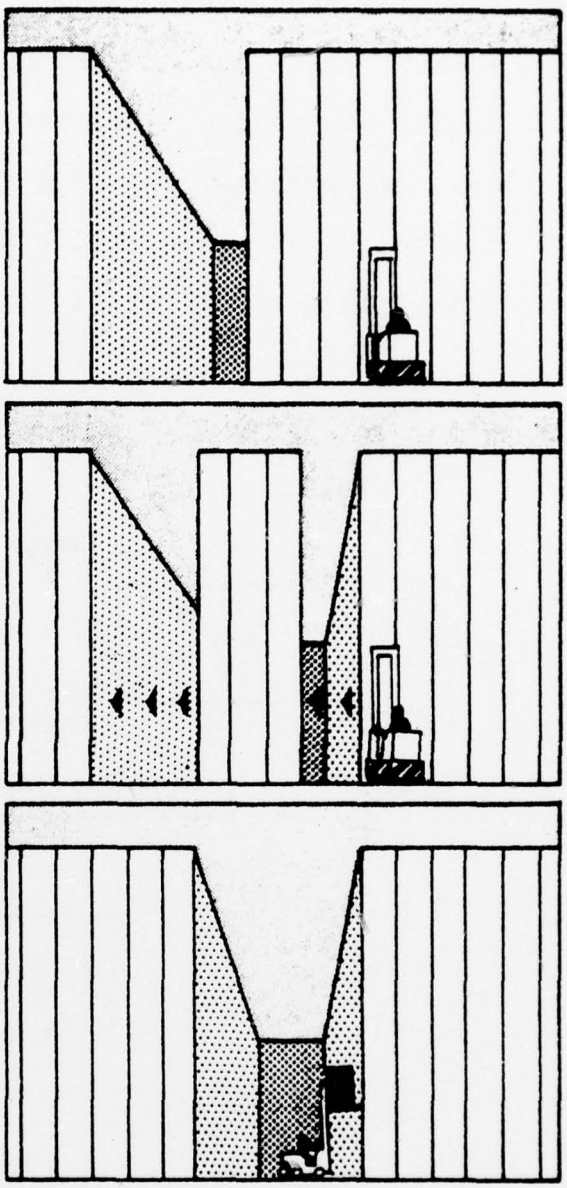


Fig.2-2: The Trans-Robot Shelves

These shelves house a few hundred high-turnover panel items. Since the material has undergone protective treatment, it is channelled in one direction only, to avoid getting damaged. Withdrawal is thus on a first-in, first-out basis. Each corridor serves one item only, and the crates progress from the loading end to the withdrawal end by means of robots. These shelves have enormous capacity, holding more than 30,000 large crates. (Source: [F1])

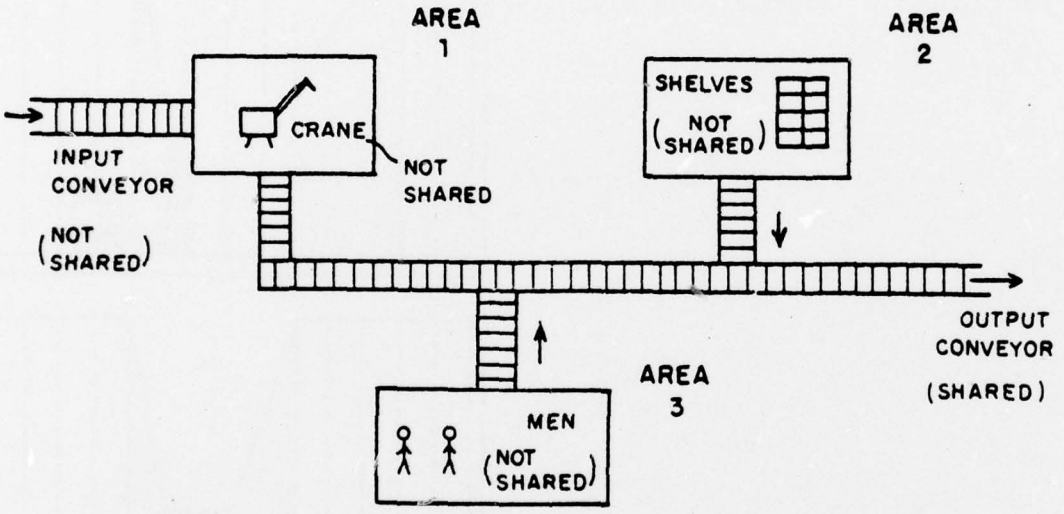
Fig.2-3: Compact Warehouse



The storage volume of these large shelves (about 50,000 cubic metres) enables 6,000 tonnes of particularly large, low-turnover material to be stocked. The shelves run on rails so that corridors are created on request only at the moment when the fork-lift trucks need to load or withdraw containers. (A safety system stops operations whenever persons or material are located in the temporary corridors.) Source: [F1].

The servicing of daily orders, and the replenishment of stocks, makes use of various resources in each area, such as cranes, conveyors, men, and shelf space. These resources may be particular to a given area (such as shelf space) or may be shared by several areas (such as a conveyor that passes through different areas: see Fig.2-4).

Fig.2-4: Examples of Various Resources



Naturally, these resources have limits on their capacity. Upon reviewing the situation at the warehouse, in January 1977, we found the salient characteristics of the resource

management problem to be as follows:

1. There are several different storage areas, each with several container-types, leading to 16 different storage-types*.

2. Each storage-type uses several resources, some of which are shared with other storage-types. there are 24 constrained resources*, such as:

   1. Storage Capacity

   2. Crane Capacity

   3. Conveyor Capacity

   4. Manual Picking Capacity

3. There were 60,000 Part-Numbers* assigned to the various storage-types on the basis of criteria that were long since outdated -- demand patterns and the Warehouse operations had changed considerably.

The net effect of these factors was bottlenecks in several resources, yet much spare capacity in others. This meant that while in some storage-types the daily demand (or

---

*The figures given here correspond to those areas of the warehouse within the scope of our project.

storage requirements) could not be met, in other storage-types equipment was lying idle**. Keeping in mind these problems, as well as the future operating requirements of the warehouse, the aims of our project were set down as:

1. "Get rid of the bottlenecks". (Improve the current allocation as quick as possible.)

2. Develop a method for reviewing the situation (say) every 3 months, and making necessary reallocations (Periodic Review).

3. Develop a rationale for allocating storage to New Part-Numbers (note that these are not replenishment stocks for existing Parts, but Parts never before stocked, e.g. for a new car model).

2.3 FORMAL STATEMENT OF PROBLEM

In this section we develop a formal model of the Storage Allocation and Resource Management problem, and indicate the factors that make a good solution (i.e. one that fulfills the aims above) difficult to find. Although we will state our model in terms of the Warehouse above, the reader will see that our model generalizes to other large

---

**Due to the special design or fixed nature of the equipment in each area, it is not generally possible to transfer capacity from one area to another.

systems (see Chapter 3, Chapter 6).

2.3.1 Notation

It is assumed that the reader has read the description of our notational conventions (following the Table of Contents). This will eliminate the need for minor definitions throughout the Chapter.

2.3.2 Problem Formulation

Let there be I Items (Part-Numbers) to be allocated in S Storage-types, such that R Resource-usage constraints are satisfied. To facilitate recall, we will use the indices i, s, or r, to represent the i^{th} item, s^{th} storage-type, or r^{th} resource.

2.3.2.1 Item Allocation - The total quantity of item i is Q^i, and its other characteristics (demand, weight, volume, etc.) are represented by a data vector $\underline{d}^i$. For each item a S-dimensional decision $\underline{x}^i$ needs to be taken, where x^i_s is the quantity of item i allocated to storage s. We will often refer to $\underline{x}^i$ as an <u>allocation</u> of item i.

2.3.2.2 Resource Usage - A given allocation for an item, along with the item's data characteristics (as above) will result in the use of various resources (e.g. storage space, crane-time, etc.). We define the resource Usage function $\underline{u}^i(\underline{d}^i,\underline{x}^i)$ to be an R-dimensional vector function such that $u^i_r(\underline{d}^i,\underline{x}^i)$ is the usage of the r^{th} resource by an item with data $\underline{d}^i$, when its allocation is $\underline{x}^i$. (The calculation of $\underline{u}(.,.)$ obviously depends on the "operating rules" of the Warehouse. For generality, we let these rules be defined for different items, hence the superscript i on $\underline{u}$ above.)

2.3.2.3 Total Allocation And Total Usages - The allocation of all items will be represented by the vector

$$\underline{x} \triangleq [\ (\underline{x}^1)',(\underline{x}^2)',\ldots,(\underline{x}^I)'\ ]'$$

The total resource usage by an allocation of all items is

$$[2.3.1] \qquad \underline{u}(\underline{x}) \triangleq \sum_{i=1}^{I} \underline{u}^i(\underline{d}^i,\underline{x}^i)$$

We will often refer to $\underline{u}$ or $\underline{u}^i$ as "usage vectors".

2.3.2.4 Constraints On Usages - The R-dimensional vector of constraints on the resource usages will be denoted by $\underline{c}$, that is c_r = value of constraint on usage of resource r.

2.3.2.5 Statement Of General Problem - Let $\underline{e}$ be the S-dimensional vector with each component equal to unity, i.e. $\underline{e} \triangleq [1,1,\ldots,1]'$. Then the Storage Allocation and Resource Management Problem can be stated as the General Problem

[2.3.2] (GP): Find $\underline{x} = [\,(\underline{x}^1)',\ldots,(\underline{x}^I)']'$

$$\text{such that} \quad \underline{e}'\underline{x}^i = Q^i \qquad \text{(I equations)}$$

$$\text{and} \quad \underline{x}^i \geq 0 \qquad \text{(S x I equations)}$$

$$\text{and} \quad \underline{u}(\underline{x}) \leq \underline{c} \qquad \text{(R equations)}$$

Note that the decision $\underline{x}$ consists of S x I components.

2.3.3 Comment On Feasibility Versus Optimality

The astute reader will already have noticed that the problem (GP), as formulated, only involves looking for a feasible solution; no notion of optimality has been stated. Some comments on the reasons for this are in order here.

The first reason for looking only for a feasible solution is that the problem is so complex (see next section) that even a feasible solution is hard to find. Thus we are satisfied if we can generate such a solution. The second, more satisfactory, reason derives from the warehouse management's objectives, which are: to keep the warehouse operational, irrespective of the relative uses of the resources, provided these usage levels are within the

limits laid down by management.* From an economic point of view too, these objectives have an explanation: the major warehouse-equipment has already been installed, and the capacities are non-transferable (sec.2.2). Further, the day-to-day operating cost of the warehouse is relatively indifferent to what equipment is being used. Thus there is no clear tradeoff between (say) using 10 more minutes of crane-time in one area versus using 2 more containers of storage in another. Hence no criterion for minimization can be stated, and all feasible solutions are equally palatable.

The fact that we look only for feasible solutions does not restrict the applicability of our approach to the special problem above. In Chapter 3 we give examples of how our method can be used in several types of optimization applications.

2.4 FACTORS CONTRIBUTING TO COMPLEXITY OF PROBLEM

Several factors make (GP) a complex problem, not amenable to standard techniques. These factors will now be discussed.

---

*We are assuming that these limits already include some margin of safety. This point will be discussed further in Chapter 5, when we study the practical aspects of our system.

2.4.1 Immense Size

In the warehouse we have I=60,000, S=16, and R=24. This leads to a decision vector of approximately one million components!

2.4.2 Part-Data And Usage Functions

The diversity of Part-Data (frequency of demand, quantity demanded, weight, volume, etc.) and the dependence of the usage functions on the physical operation of the warehouse, leads to usage functions which can be discontinuous and/or nonlinear and/or nonconvex. For example if a quantity Q of an item is to be "picked", and fits entirely in one container, then one crane operation can retrieve the container. However, for larger Q two containers may need to be retrieved, requiring two crane operations. Thus crane-time usage can be a discontinuous function of the quantity demanded.

2.4.3 Incoming New Part-Numbers

In addition to the 60,000 items in the warehouse, there are 30-50 New items arriving every day. As mentioned in sec.2.2, these are not replenishment stocks, but entirely new items. Hence 30-50 new allocations $\underline{x}^i$ have to be made every day, and clearly we would like to make "reasonable" decisions without re-solving the whole problem (GP) for the

combined set of old and new items. By "reasonable" we mean a decision that will remain valid in the long run, that is we should not have to re-allocate these New Parts in the near future.

2.4.4 Shortcomings Of Standard Techniques

Initially, we might have been tempted to try Linear Programming or Integer Programming techniques, using linearized approximations where necessary. In view of the above remarks we see that these would suffer from other major disadvantages: first, the decision vector of one million components would lead to an astronomical program; and second, these methods would not lead to any strategy for allocating the new parts, short of re-solving the problem for each new set of parts.

In the second part of this Chapter we shall see how an appropriate reformulation of the problem (GP) leads us to better solution tools.

Part II: Decentralized Solution Techniques

2.5 INTRODUCTION TO PART II

In the preceding sections we described the Resource Management problem in a Large Warehouse, discussed the reasons which make it a hard problem, and stated why we are satisfied with any feasible solution to the problem. In this Part we develop the beginnings of a solution technique for the problem. Our method is to set up an "Artificial" optimization problem, and to use Lagrange Multipliers to simplify the solution to this problem, through decomposition into a large number of smaller (much easier) problems. We will show that solving these smaller problems also gives an approach for the New Parts problem (sec.2.4.3). Then we will describe conventional iteration methods which, using the smaller problems, might be able to solve the Resource Management problem.

The use of Lagrange Multipliers for decomposition of large problems is well known (see discussion in sec.2.7). In general however, such methods suffer from some severe disadvantages, the most devastating of these being the possibility of "duality gaps". This means that there may not exist multipliers which generate an optimum to the Artificial problem. A detailed discussion of this and other disadvantages is given at the end of sec.2.7.

In the remainder of this Chapter we show the reader the advantages of the decentralized approach. This will motivate us to investigate whether we can overcome the disadvantages mentioned above. It shall be that very investigation which, in Chapters 3 and 4, will lead to the main contributions of this work.

2.6 THE ARTIFICIAL PROBLEM AND LAGRANGE MULTIPLIERS

2.6.1 The Artificial Problem

In order to put the problem (GP) [2.3.2] in conventional optimization terms we formulate the "Artificial" Problem

$$[2.6.1] \qquad (AP): \quad \max J(\underline{x}) \triangleq \sum_{i=1}^{I} \underline{e}'\underline{x}^i$$

$$[2.6.2] \qquad \text{subject to} \quad \underline{x}^i \geq 0 \quad \text{each } i$$

$$[2.6.3] \qquad Q^i - \underline{e}'\underline{x}^i \geq 0 \quad \text{each } i$$

$$[2.6.4] \qquad \underline{c} - \underline{u}(\underline{x}) \geq 0$$

In other words, we want to maximize the total quantity allocated, subject to the resource usage constraint, the non-negativity constraint, and the fact that at most we can allocate the quantity we have of each item. If a feasible solution exists to (GP), then the maximum value of (AP) will be $\sum_{i=1}^{I} Q^i$. (Notice the analogy with the Artificial variable technique of Linear Programming where, if a

feasible solution exists to the original problem, then the optimal value of the Artificial Problem is zero. This point will be amplified in Chapter 3.)

2.6.2 Lagrange Multiplier Method

We tackle (AP) by the familiar Lagrange Multiplier method. Let $\underline{\lambda}$ be a R-dimensional vector of Lagrange Multipliers. We write the Lagrangean associated with (AP) as

[2.6.5] $$L(\underline{x},\underline{\lambda}) = J(\underline{x}) - \underline{\lambda}'[\underline{u}(\underline{x})-\underline{c}]$$

For each i, let X^i be the set of $\underline{x}^i$ which satisfy [2.6.2] and [2.6.3], and let X be the set of $\underline{x}$ such that $\underline{x}^i \in X^i$ for each i. Then there is the following "Saddle Point Theorem" (see for example Lasdon [L1]):

If there exist $(\underline{x}^*,\underline{\lambda}^*)$ with $\underline{x}^* \in X$ and $\underline{\lambda}^* \geq 0$ such that

[2.6.6] $$L(\underline{x},\underline{\lambda}^*) \leq L(\underline{x}^*,\underline{\lambda}^*) \leq L(\underline{x}^*,\underline{\lambda})$$

for all $\underline{x} \in X$ and $\underline{\lambda} \geq 0$, then $\underline{x}^*$ solves the problem (AP).

[]

The power of the above result lies in the fact that it does <u>not</u> depend on the form of the functions $J(\underline{x})$ and $\underline{u}(\underline{x})$, nor on the form of the set X [L1]. An alternative view of [2.6.6], which will be used below, is to say that

[2.6.7] $$\underline{x}^* = \arg \max_{\underline{x} \in X} L(\underline{x},\underline{\lambda}^*)$$

[2.6.8] $$\underline{\lambda}^* = \arg \min_{\underline{\lambda} \geq 0} L(\underline{x}^*,\underline{\lambda})$$

2.6.3 Decentralization Of Decisions

A key point to note is that for given $\underline{\lambda}$ the problem [2.6.7] can be decentralized since

$$[2.6.9] \qquad \max_{\underline{x} \in X} L(\underline{x},\underline{\lambda}) = \underline{\lambda}'\underline{c} + \sum_{i=1}^{I} \max_{\underline{x}^i \in X^i} \{\underline{e}'\underline{x}^i - \underline{\lambda}'\underline{u}^i(\underline{d}^i,\underline{x}^i)\}$$

Thus, for given $\underline{\lambda}$, the decision for each item i can be taken independently of the others, by solving the (much simpler) Individual Problem

$$[2.6.10] \qquad (IP): \quad \max_{\underline{x}^i \in X^i} \{\underline{e}'\underline{x}^i - \underline{\lambda}'\underline{u}^i(\underline{d}^i,\underline{x}^i)\}$$

We note here that in Chapter 4 we shall further simplify (IP). In Chapter 5 we will show how (for the warehouse case) we were able to find an easy solution to this individual problem, and also show that solving (IP) gives us an approach for the New Parts problem.

2.7 KNOWN SCHEMES FOR FINDING OPTIMAL MULTIPLIERS

We see above that a given $\underline{\lambda}$, through (IP), leads to an allocation of all items, say $\underline{x}(\underline{\lambda})$, and corresponding total resource usages $\underline{u}(\underline{x}(\underline{\lambda}))$. We can therefore think of $\underline{u}$ as a function of $\underline{\lambda}$, and we shall henceforth write it simply as $\underline{u}(\underline{\lambda})$. The problem then, is to find the $\underline{\lambda}^*$ in [2.6.8], for then from (IP), [2.6.7], and [2.6.6] we know that $\underline{x}(\underline{\lambda}^*)$ and $\underline{u}(\underline{\lambda}^*)$ are optimal.*

---

*For the moment we assume such a $\underline{\lambda}^*$ exists. This issue will be briefly discussed in the next section, and answered in detail in Chapter 3.

Arrow and Hurwicz[A1] observed that [2.6.7] and [2.6.8] suggest an iterative scheme of the form

$$[2.7.1] \qquad \underline{x}^{k+1} = \arg \max_{\underline{x} \in X} \; L(\underline{x}, \underline{\lambda}^k)$$

$$[2.7.2] \qquad \underline{\lambda}^{k+1} = \arg \min_{\underline{\lambda} \geq 0} \; L(\underline{x}^{k+1}, \underline{\lambda})$$

They noted that [2.7.1] can be solved using (IP) with given $\underline{\lambda}=\underline{\lambda}^k$, and an alternative to solving [2.7.2] exactly at each iteration is to let $\underline{\lambda}^k$ be changed in the direction of the negative gradient of L with respect to $\underline{\lambda}$, which is then simply

$$[2.7.3] \qquad \lambda_j^{k+1} = \begin{cases} \lambda_j^k \text{ if } \lambda_j^k = 0 \text{ and } u_j(\underline{\lambda}^k) < c_j \\ \lambda_j^k + a_j\{u_j(\underline{\lambda}^k) - c_j\} \text{ otherwise} \end{cases}$$

where a_j is some constant. The scheme [2.7.1],[2.7.3] then has an intuitively appealing economic interpretation. A "central co-ordinator" chooses a set of "prices" $\underline{\lambda}$, after which the items i find their optimal decisions $\underline{x}^i$ for this $\underline{\lambda}$. The central co-ordinator then looks at the total resource usages and adjusts the prices to increase the cost of over-used resources, and decrease the cost of under-used resources (but never making any cost negative); in other words he adjusts prices according to excess demand. This use of decentralization in Resource Allocation problems is well known [A1,E1,G1,L1,S2], and arises out of the additive nature of the objective function [2.6.1] and the resource usage functions (see [2.6.4] along with [2.3.1]).

We have reduced via this means an optimization problem involving S x I (=one million) variables to an optimization problem with R (=24) variables plus a set of I (=60,000) decoupled and relatively simple problems. However, we must overcome three additional difficulties:

1. The decomposition and iteration method described above falls in the general category of "dual" methods [G1]. A major shortcoming of these methods is the existence of "duality gaps" [G3,L1] -- this means that although an optimal value of the Artificial Problem exists, no pair ($\underline{x}^*,\underline{\lambda}^*$) exists which satisfies the Saddle Point Condition [2.6.6]. Thus, no $\underline{\lambda}$ can achieve the optimum value using the schemes given above. (The name "duality gap" derives from the fact that there is a gap between the actual maximum and the maximum achievable using the dual method.)

2. Even if no duality gap exists, convergence of [2.7.3] is guaranteed only when strict conditions (such as convexity/continuity) hold on the Payoff Function and Resource usage Functions [A1,Z1] -- conditions which certainly do not hold in our problem.

3. Convergence can be very slow even given the above conditions.

We therefore look for an improved scheme. At this stage the reader might well ask what motivated us to pursue this line of attack in the face of such difficulties, so we pause a moment to gather our thoughts on this issue.

2.8 ADVANTAGES OF DECENTRALIZED APPROACH

The reasons why we chose to pursue this solution technique are manifold:

1. It makes possible the solution of a large intractable problem, by reducing it to a number of smaller problems.

2. Suppose we are able to find an efficient iteration technique, and use it to generate a solution $\underline{\lambda}^*$, with corresponding allocation $\underline{x}(\underline{\lambda}^*)$. When demand characteristics have changed slightly over some months, we still expect $\underline{\lambda}^*$ to be a good starting point for iterations to find a new solution. Hence the Periodic Review problem (sec.2.2) can be solved very efficiently each time.

3. Given a set of multipliers $\underline{\lambda}^*$, the New Parts problem (sec.2.4.3) can be reduced to solving (IP) for each new part. This problem is relatively easy and can be solved separately for each new part. Hence the allocation of new parts is (through $\underline{\lambda}^*$) made independent of the rest of the parts in the

warehouse. (The practical implementation of this scheme demands some care, and is described in Chapter 5.)

4. The economic interpretation of the scheme (sec.2.7) makes it appealing to Managers, who readily understand it. Hence they prefer it to other schemes which give them no insight as to the rationale behind a particular allocation.

Thus, encouraged by the positive aspects of the decentralized approach, we proceed to resolve the problems mentioned in the preceding section. This will be the subject of the next two Chapters.

CHAPTER 3

AN EXISTENCE THEOREM FOR OPTIMAL MULTIPLIERS

NOTE

The results in this Chapter are of interest in their own right. For this reason it is written so that it can be read independently of the other Chapters.

3.1 INTRODUCTION

3.1.1 Motivation

The task of finding feasible solutions to large allocation problems can often be quite complex. Such a task occurs in many situations (see sec.3.1.3). In this work we present a technique for accomplishing this task. In the domain of Linear Programming, the artificial-variable method is used for finding initial feasible solutions, by setting up an Artificial Optimization problem. Analogously, our technique is to set up an Artificial Problem, and use

Lagrange Multipliers to simplify the solution to this problem, through decomposition into several easier problems. This effectively reduces the allocation problem to one of finding suitable multipliers.

The contribution of this Chapter is to give a firm theoretical basis for our technique. The main theorem gives simple conditions which guarantee the existence of optimal multipliers for the Artificial Problem. Although the use of Lagrange Multipliers for allocation problems is well known, such existence results have previously been given only under strict conditions (e.g. convexity, continuity). Our Theorem justifies the use of this technique in many cases where the resource usage functions are nonconvex and/or discontinuous, and the domains of the decision variables are nonconvex.

An iterative algorithm for solving the Artificial Problem, using our technique, is described in the next Chapter. In that Chapter we also show that this algorithm has proveable convergence properties, and a quadratic rate of convergence. It is thus expected to be of practical value; indeed, in Chapter 5 we describe one instance of its use in a large warehouse.

3.1.2 Abstract Statement Of Problem

In Chapter 2 we described the Resource Management problem in a very large FIAT spare-parts warehouse. In this section we state the problem in abstract terms, so that the reader can appreciate the generality of our work.

Consider the problem of allocating several resources among a set of independent activities. Specifically, let

A^i, $i \in \{1,2,\ldots,I\}$	be the ith Activity.
$x^i \in S^i$	be the strategy (decision) of A^i, with S^i a discrete strategy set with a finite number of elements.
$\underline{u}^i(x^i) \in E^R$	be an R-dimensional resource Usage vector, where $u^i_k(x^i)$ is the amount of resource k used by A^i when strategy x^i is employed (each $u^i_k(x^i) \geq 0$).
$\underline{x} \hat{=} [x^1, x^2, \ldots, x^I]'$	be an Allocation (each $x^i \in S^i$).
$\underline{u}(\underline{x}) \hat{=} \sum_{i=1}^{I} \underline{u}^i(x^i)$	be the total resource Usage by an allocation.
$\underline{c} \in E^R$	be the vector of Constraints on the usages (c_k=limit on usage of resource k)

Typically, we have in mind problems where I (the number of activities) is in the tens of thousands, R (the number of resources) is in the range 10 to 50, and the number of elements in each strategy set is in the hundreds.

Usually, one has the following "Optimization Problem":

[3.1.1] (OP): $\max_{\underline{x}} H(\underline{x})$ a real valued function

subject to $x^i \in S^i$ for each i

and* $\underline{u}(\underline{x}) \leq \underline{c}$

In this Chapter we consider the "Feasibility Problem":

[3.1.2] (FP): Find $\underline{x}$ such that

$x^i \in S^i$ for each i

and $\underline{u}(\underline{x}) \leq \underline{c}$

3.1.3 The Utility Of The Feasibility Problem

The solution to (FP) is useful for a variety of reasons:

1. In the case of the problem (OP), there may exist iteration methods which improve a given <u>feasible</u> solution. In this case a completely different algorithm may be required to generate an initial feasible solution. (See [M1] for an example of this.)

---

*All vector inequalities are to be interpreted componentwise. (See notation conventions following Table of Contents.)

2. In the case of an operational system, it may be required only to keep it operational, i.e. to find an allocation such that no constraints are violated, and to maintain this condition during the operation of the system. Examples of this are Parts-allocation in a large warehouse (see Chapter 2) and file-allocation in a distributed computer network (Chapter 6).

3. In many system design problems, a unique criterion to be minimized cannot be formulated. In such cases, one approach is to define certain constraints, find a solution to satisfy these, and then look at the other performance parameters of the resulting design. This approach has been advocated by Chang [C1] for the general Distributed Computer System design problem. It has also been proposed for solving multicriteria decision problems [L3].

When I is very large, the functions $u^i(.)$ are not linear, and each S^i contains a substantial number of points, (FP) may itself be a difficult problem.* Our method is to

---

*We are assuming ofcourse that, for each i, S^i does not contain an $\hat{x}^i$ such that $u^i(\hat{x}^i)=0$. For if this were the case, we could set $x^i=\hat{x}^i$ for each i and the problem (FP) would be trivial. In the warehouse problem (Chapter 2) for instance, each part must be allocated somewhere, so that there is no x^i for which $u^i(x^i)=0$.

set up an "Artificial" Optimization problem, and to use Larange Multipliers to simplify the solution to this problem, through decomposition into I (easier) problems. The decomposition technique is well known [A1,E1,G1,L1,S2]. In general however such methods suffer from three disadvantages, which are briefly**: (i) The existence of "duality gaps", which means that there may not exist multipliers which generate an optimum to the Artificial Problem, (ii) Convergence of solution algorithms requires stringent conditions on the functions $\underline{u}^i(.)$ and the sets S^i, and (iii) Slow convergence of such algorithms.

In this Chapter we give simple conditions which guarantee the existence of optimal multipliers for the Artificial Problem, under very general conditions on the $\underline{u}^i(.)$ and S^i. A solution algorithm for the Feasible Problem (above) is described in Chapter 4. We mention that this algorithm has guaranteed convergence properties under stricter, but still fairly general conditions on the $\underline{u}^i(.)$. It is being successfully used in a FIAT spare-parts warehouse with I=60,000 , each S^i has 16 points, the functions $\underline{u}^i(.)$ are nonlinear and/or discontinuous, and the number of resources (R) is 24 (Chapter 5).

**These points are repeated here to keep this Chapter self-contained. See sec.2.7 for details and references.

3.2 THE ARTIFICIAL PROBLEM

3.2.1 Analogy With Linear Programming

An artificial problem is used in Linear Programming (LP) to generate an Initial Basic Feasible Solution (IBFS). A brief review of the technique follows (see [D1] or [L5] for details).

Consider an LP problem whose constraints are:

[3.2.1] (LP1): $A\underline{x} = \underline{b}$ where A is a matrix, and $\underline{b} \geq 0$

$\underline{x} \geq 0$.

(Through use of slack variables it is always possible to express inequality constraints in this form.) In order to find an IBFS to (LP1) we consider the (artificial) problem

[3.2.2] (LP2): $\min \sum_i z_i$

subject to $A\underline{x} + \underline{z} = \underline{b}$

$\underline{x} \geq 0$

$\underline{z} \geq 0$.

Clearly, if a feasible solution exists to (LP1), then the minimum value of (LP2) will be zero, and conversely. But now, (LP2) has an obvious IBFS ($\underline{x}=0$, $\underline{z}=\underline{b}$), and can be solved using the simplex method. The optimum to this problem, if zero, will result from a value of $\underline{x}$ which can now be used as an IBFS for (LP1).

Thus we see that the above technique (called the Artificial Variable method in LP) sets up an "artificial" problem whose optimum is any feasible solution to the original problem. This is essentially our approach below.

3.2.2 Formulation Of Artificial Problem

[3.2.3] Definition: (Augmented Strategy Sets) To each strategy set S^i, let us adjoin a decision θ, with

$$\underline{u}^i(\theta) \triangleq 0 \ .$$

We shall also use

$$S_+^i \triangleq S^i \cup \{\theta\}$$

$$S \triangleq S^1 \times S^2 \times \ldots \times S^I$$

$$S_+ \triangleq S_+^i \times S_+^2 \times \ldots \times S_+^I$$

so that $\underline{x} \in S \iff x^i \in S^i$ for each i ; similarly $\underline{x} \in S_+$. []

[3.2.4] Remark: We can assume, without loss of generality, that $\theta \notin S^i$. For if $\theta \in S^i$ for some i, then set $x^i = \theta$, remove x^i from $\underline{x}$, and then solve the remaining problem below. Also see footnote in sec.3.1.3. []

We can now state the Artificial Problem:

$$[3.2.5] \qquad (AP): \quad \max_{\underline{x}} J(\underline{x}) \triangleq \sum_{i=1}^{I} p^i(x^i)$$

$$\text{such that } \underline{x} \in S_+$$

$$\text{and } \underline{u}(\underline{x}) \leq \underline{c}$$

where

$$p^i = 0 \quad \text{if } x^i = \theta$$

$$p^i = P^i \quad \text{if } x^i \in S^i \quad (P^i \text{ are positive constants}).$$

Now let $J^* \triangleq \sum_{i=1}^{I} P^i$. In analogy with the LP example, we see that if a feasible solution exists to (FP), then the maximum value of (AP) will be J^*, and conversely, any $\underline{x}$ achieving J^* in (AP) must be a solution to (FP).

3.2.3 Lagrange Multipliers And Decentralized Solutions*

We tackle (AP) by the familiar Lagrange Multiplier method. Let $\underline{\lambda}$ be a R-dimensional vector of Lagrange Multipliers. We write the Lagrangean associated with (AP) as

$$[3.2.6] \qquad L(\underline{x},\underline{\lambda}) = J(\underline{x}) - \underline{\lambda}'[\underline{u}(\underline{x})-\underline{c}]$$

---

*In order to make this Chapter self-contained, the next few paragraphs are duplicated from Chapter 2. There will be no further repetition of any material in this Chapter.

Then there is the following "Saddle Point Theorem" (see for example Lasdon [L1]):

If there exist $(\underline{x}^*,\underline{\lambda}^*)$ with $\underline{x}^* \in S_+$ and $\underline{\lambda}^* \geq 0$ such that

[3.2.7] $\quad L(\underline{x},\underline{\lambda}^*) \leq L(\underline{x}^*,\underline{\lambda}^*) \leq L(\underline{x}^*,\underline{\lambda})$

for all $\underline{x} \in S_+$ and $\underline{\lambda} \geq 0$, then $\underline{x}^*$ solves the problem (AP).

[]

The power of the above result lies in the fact that it does <u>not</u> depend on the form of the functions $J(\underline{x})$ and $\underline{u}(\underline{x})$, nor on the form of the set S_+ [L1]. An alternative view of [3.2.7], which will be used below, is to say that

[3.2.8] $$\underline{x}^* = \arg \max_{\underline{x} \in S_+} L(\underline{x},\underline{\lambda}^*)$$

[3.2.9] $$\underline{\lambda}^* = \arg \min_{\underline{\lambda} \geq 0} L(\underline{x}^*,\underline{\lambda})$$

We note further that for given $\underline{\lambda}$ the problem [3.2.8] can be decentralized since

[3.2.10] $$\max_{\underline{x} \in S_+} L(\underline{x},\underline{\lambda}) = \underline{\lambda}'\underline{c} + \sum_{i=1}^{I} \max_{x^i \in S_+^i} \{p^i(x^i) - \underline{\lambda}'\underline{u}^i(x^i)\}$$

Thus, for given $\underline{\lambda}$, the decision for each item i can be taken independently of the others, by solving the (much simpler) Individual Problem

[3.2.11] $$\text{(IP):} \quad \max_{x^i \in S_+^i} \{p^i(x^i) - \underline{\lambda}'\underline{u}^i(x^i)\}$$

(This problem could be solved, at worst, simply by enumerating the points of S_+^i).

For given $\underline{\lambda}$ we shall denote the solution to [3.2.8] by $\hat{\underline{x}}(\underline{\lambda})$, that is

[3.2.12] $$\hat{\underline{x}}(\underline{\lambda}) \in \arg \max_{\underline{x} \in S_+} L(\underline{x},\underline{\lambda})$$

Since the solution to [3.2.12] is relatively simple for a given $\underline{\lambda}$, our aim is to determine

1. Whether <u>there exists</u> a $\underline{\lambda}^*$ such that $\hat{\underline{x}}(\underline{\lambda}^*)$ achieves the maximum in (AP) ?

2. If such a $\underline{\lambda}^*$ exists, how can it be found ?

As already mentioned, the second question will be tackled by us in Chapter 4. In the rest of this Chapter we address the first question. Our theorem will be quite different from the Saddle Point Theorem.

3.3 MAIN THEOREM

The question of existence of an optimal $\underline{\lambda}$ for a given problem has, in general, only been answered in the literature under certain convexity conditions [G3,L1,L4]. In this section we take advantage of the structure of a large allocation problem to give a far more general result.

[D3.3.1] Definition: We define the R-dimensional vector $\underline{\Delta}$, whose k^{th} component represents the largest change in the usage of the k^{th} resource, that can be caused by a single activity A^i (with its decisions restricted to its strategy set S^i):

$$\Delta_k \hat{=} \max_i \; \max_{x_1^i \in S^i, x_2^i \in S^i} |u_k^i(x_1^i) - u_k^i(x_2^i)| \quad []$$

[T3.3.2] Theorem 3-I:

If there exists an $\underline{\bar{x}} \in S_+$ with $J(\underline{\bar{x}}) = J^*$ and $\underline{u}(\underline{\bar{x}}) \leq \underline{\bar{c}} - \hat{a}\underline{\Delta}$, where $\hat{a} \hat{=} (R-1)/2$, then there exists a $\underline{\lambda}^* \geq 0$ and an $\underline{\hat{x}}(\underline{\lambda}^*)$ as in [3.2.12] such that $J(\underline{\hat{x}}(\underline{\lambda}^*)) = J^*$ and $\underline{u}(\underline{\hat{x}}(\underline{\lambda}^*)) \leq \underline{\bar{c}}$, that is, $\underline{\hat{x}}(\underline{\lambda}^*)$ solves (AP). []

For a large problem with (say) several thousand activities using each resource, we would expect $\hat{a}\underline{\Delta}$ to be very small in comparison with $\underline{\bar{c}}$. In that case we can give the following

Interpretation of Theorem 3-I:

If, for a slightly tighter set of limits, the original problem is still feasible, then there will exist a $\underline{\lambda}^* \geq 0$ such that the (decentralized) solution $\underline{\hat{x}}(\underline{\lambda}^*)$ will also be feasible for the original problem. (This interpretation will be illustrated by a diagram in sec.3.5.4) []

Remarks: The importance of our theorem is threefold --

1. We have given conditions under which there will be no duality gap in the Artificial Problem.

2. Our conditions require no convexity and/or continuity and/or linearity assumptions. In comparison to existing conditions they are extremely mild, and likely to be true for most large systems (since $\hat{a}\underline{\Delta}$ is very small in comparison with $\underline{\bar{c}}$, as explained above).

3. If each S^i has N components, our theorem justifies replacing the IxN Integer Programming problem for $\underline{x}$ by the R-dimensional Nonlinear Programming problem of finding a suitable $\underline{\lambda}$. For instance, in the case of the FIAT warehouse problem (Chapter 2), IxN has approximately one million components, whereas R has only 24.

3.3.1 Outline Of Proof Of Theorem

Since the proof of the above Theorem is quite long, we give the reader a flavour for what needs to be accomplished.

3.3.1.1 Motivation - Let us consider the R+1 dimensional space of payoff versus resource usage for (AP). In this space consider the set

$$\mathcal{J} \triangleq \{ \begin{bmatrix} J \\ \underline{u} \end{bmatrix} \mid J=J(\underline{x}),\ \underline{u}=\underline{u}(\underline{x}),\ \text{for } \underline{x} \in S_+ \}$$

The problem [3.2.12] finds a solution $\hat{\underline{x}} \in S_+$ such that

$$J(\hat{\underline{x}})-\underline{\lambda}'[\underline{u}(\hat{\underline{x}})-\underline{c}] \geq J(\underline{x})-\underline{\lambda}'[\underline{u}(\underline{x})-\underline{c}] \text{ for all } \underline{x} \in S_+$$

or, with $\begin{bmatrix} \hat{J} \\ \hat{\underline{u}} \end{bmatrix} \triangleq \begin{bmatrix} J(\hat{\underline{x}}) \\ \underline{u}(\hat{\underline{x}}) \end{bmatrix}$ we can write this as

$$[1,\underline{\lambda}'] \begin{bmatrix} \hat{J} \\ \hat{\underline{u}} \end{bmatrix} \geq [1,\underline{\lambda}'] \begin{bmatrix} J \\ \underline{u} \end{bmatrix} \text{ for all } \begin{bmatrix} J \\ \underline{u} \end{bmatrix} \in \mathcal{J}$$

which is a support hyperplane for the set $\mathcal{J}$, passing through the point $\begin{bmatrix} \hat{J} \\ \hat{\underline{u}} \end{bmatrix} \in \mathcal{J}$. Thus, solutions found using [3.2.12] can only lie on these support planes.

The source of duality gaps is that (if $\mathcal{J}$ is nonconvex, say) there may exist solution points on the nonconvex boundary of $\mathcal{J}$ which are not touched by any support plane of $\mathcal{J}$.

Our approach is to show that, although such "unreachable" points may exist, the $\hat{a}\underline{\Delta}$ condition in the Theorem assures us that there exists some other "reachable" point which is also an optimum for (AP). We do this as follows.

First we look at the projection of a subset of $\mathcal{J}$ onto the $\underline{u}$-space. Specifically, we look at the set U^* of all $\underline{u}$ such that $\begin{bmatrix} J \\ \underline{u} \end{bmatrix} \in \mathcal{J}$, with $J=J^*$. This will be illustrated by an example.

3.3.1.2 Simple Allocation Example -

Let I=3 (number of activities)
and R=2 (number of resources)

Suppose the activities are:

A^1 with $x^1 \in S^1 \hat{=} \{a,b\}$ and $\underline{u}^1_1(a) = [1,4]'$
$\underline{u}^1(b) = [3,3]'$

A^2 with $x^2 \in S^2 \hat{=} \{a,b\}$ and $\underline{u}^2_2(a) = [3,5]'$
$\underline{u}^2(b) = [4,3]'$

A^3 with $x^3 \in S^3 \hat{=} \{a,b,c\}$ and $\underline{u}^3_3(a) = [4,4]'$
$\underline{u}^3(b) = [5.5,3.5]'$
$\underline{u}^3(c) = [6,2]'$

The augmented strategy set S^1_+, for example, would be $S^1_+=\{a,b,\theta\}$ with $\underline{u}^1(\theta)=[0,0]'$.

The possible values of $\underline{u}(\underline{x})=\sum_{i=1}^{3} \underline{u}^i(x^i)$, *for* $\underline{x} \in S$, *are* shown in Table 3-I. For example, in the 4th row of the table, for $\underline{x}=[a,b,a]'$ we have

$$\underline{u}(\underline{x}) = \underline{u}^1(a)+\underline{u}^2(b)+\underline{u}^3(a) = [9,11]' .$$

Since, for $J(\underline{x})=J^*$ we must have $\underline{x} \in S$, the set U^* is precisely the set of values in the Table, and this set is illustrated in Fig.3-1.

By examining the values of $\underline{u}^i(x^i)$ above, we find (see definition [D3.3.1])

$$\underline{\Delta} = [2,2]' .$$

Since R=2, we have $\hat{a}=(2-1)/2 = 0.5$ so that

$$\hat{a}\underline{\Delta} = [1,1]' .$$

We shall need this value in later illustrations. []

Table 3-I: Values of $\underline{u}(\underline{x})$ for $\underline{x} \in S$ in the Example

$\underline{x}$			$\underline{u}(\underline{x})$	
x^1	x^2	x^3	$u_1(\underline{x})$	$u_2(\underline{x})$
a	a	a	8	13
a	a	b	9.5	12.5
a	a	c	10	11
a	b	a	9	11
a	b	b	10.5	10.5
a	b	c	11	9
b	a	a	10	12
b	a	b	11.5	11.5
b	a	c	12	10
b	b	a	11	10
b	b	b	12.5	9.5
b	b	c	13	8

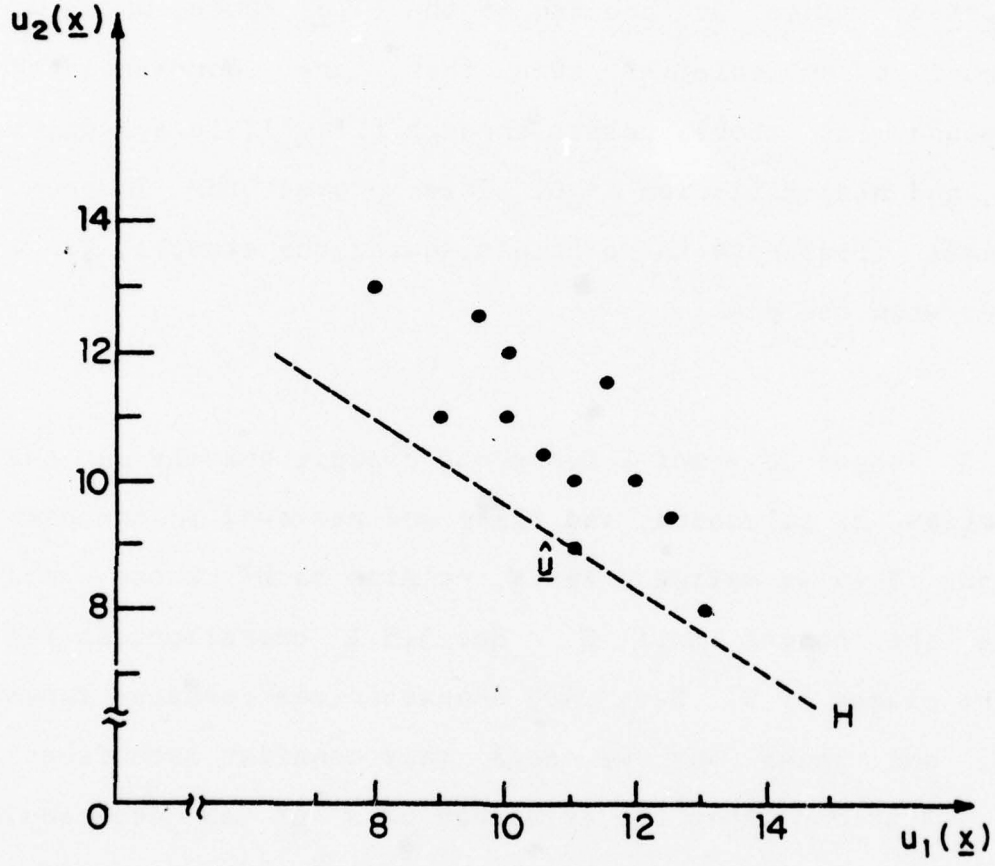


Fig.3-1: The set U* for the Example.

The set consists of all possible values of $\underline{u}(\underline{x})$, shown by heavy dots. The line H is a typical support hyperplane for the set U*, passing through the point $\hat{\underline{u}}$.

With U* as defined above, our proof shows that the $\hat{a}\underline{\Delta}$ condition in Theorem 3-I guarantees the existence of some $\underline{u}^* \epsilon U^*$ which lies on the "reachable" boundary of U*, and also has $\underline{u}^* \leq \bar{c}$. Then we proceed to the $(J,\underline{u})$ space, where we construct a suitable $\underline{\lambda}^*$ such that the corresponding hyperplane (as above) passes through $(J^*,\underline{u}^*)$, is a support for $\mathfrak{J}$, and also satisfies $\underline{\lambda}^* \geq 0$. This proves the Theorem. We shall illustrate these points, using the example, as we proceed with the proof.

3.3.1.3 Stages In Proof - Our proof depends heavily on the properties of polyhedra, and these are reviewed in the next section. Then we define a set W, related to U* above, and define its convex hull $\hat{W}$. Sec.3.5.2 characterizes the support planes of $\hat{W}$. Sec.3.5.3 characterizes certain faces of $\hat{W}$, and shows why we need only consider such faces. Sec.3.5.4 accomplishes the existence of a $\underline{u}^*$ as described above. Finally Sec.3.5.5 carries out the construction of a suitable $\underline{\lambda}^*$.

3.4 CONCEPTS RELATING TO POLYHEDRA -- A BRIEF REVIEW

Our proof relies heavily on the properties of Polyhedra. Hence a review of relevent concepts is given here. The reader should be aware that several different (but equivalent) definitions can be given. Our exposition is taken mainly from [S1].

We shall be concerned with n-dimensional Euclidean spaces E^n. For $U \subset E^n$, $\underline{u}^i \in U$, and a^i scalars, we define the conical hull of U

$$\text{cone}(U) \triangleq \{ \sum_{i=1}^{N} a^i \underline{u}^i \mid a^i \geq 0 \} \text{ for any finite } N,$$

the convex hull of U

$$\text{convex}(U) \triangleq \{ \sum_{i=1}^{N} a^i \underline{u}^i \mid a^i \geq 0, \sum_{i=1}^{N} a^i = 1 \} \text{ for any finite } N,$$

and the affine hull of U

$$\text{affine}(U) \triangleq \{ \sum_{i=1}^{N} a^i \underline{u}^i \mid \sum_{i=1}^{N} a^i = 1 \} \text{ for any finite } N.$$

The sum of two sets $U, V \subset E^n$ is

$$U+V \triangleq \{\underline{u}+\underline{v} \mid \underline{u} \in U \text{ and } \underline{v} \in V\} .$$

[D3.4.1] Definition: The solution set $P \subset E^n$ of a finite system of linear inequalities

$$P \triangleq \{ \underline{c} \mid \underline{h}^{i\prime} \underline{c} \leq b^i, \quad \underline{h}^i, \underline{c} \in E^n, \quad i=1,\ldots,N\}$$

is called a convex polyhedron (or briefly, polyhedron). []

[P3.4.2] Property: If U and V are finite point sets in E^n then

$$P = \text{convex}(U) + \text{cone}(V)$$

is a polyhedron. []

In the following statements, P will always denote a polyhedron.

[D3.4.3] Definition: In [D3.4.1], for each i the set

$$\{ \underline{c} \mid \underline{h}^i{}'\underline{c} = b^i \}$$

is called a boundary plane of P. []

[D3.4.4] Definition: The intersection of P with some of its boundary planes is called a face. (A face is itself a polyhedron. Also, if F1 is a face of P and F2 a face of F1, then F2 is a face of P.) A face consisting of exactly one point is called a vertex. A face which is different from P is called a proper face. []

[D3.4.5] Definition: $\underline{c}$ is an extreme point of P if

$$\underline{c} \in \text{convex}\{ \underline{c}^1, \underline{c}^2, \ldots, \underline{c}^N \} \text{ for } \underline{c}^i \in P$$

implies $\underline{c}^i = \underline{c}$ for all i. []

[P3.4.6] Property: Every extreme point of P is a vertex, and conversely. []

[D3.4.7] Definition: P is called bounded if there exists a

scalar e such that for all $\underline{c} \in P$, $|c_i| < e$. []

[P3.4.8] Property: Every bounded polyhedron is the convex hull of its finitely many extreme points (vertices). []

[P3.4.9] Property: P is bounded if and only if it contains no halfline. []

[D3.4.10] Definition: Any point belonging to a proper face of P is a boundary point. All other points of P are inner points. []

[D3.4.11] Definition: A boundary plane containing the entire polyhedron is called singular. (For example, if the face F is the intersection of P with the boundary plane H, then w.r.t. the polyhedron F, the plane H is a singular boundary plane.) []

[P3.4.12] Property: If $\underline{c}$ is an inner point of P, and $\underline{c}''$ an arbitrary point on the intersection of all singular boundary planes of P, then there exists an $e>0$ such that

$$\underline{c}+e(\underline{c}-\underline{c}'') \text{ and } \underline{c}-e(\underline{c}-\underline{c}'')$$

are both in P. (If there are no singular boundary planes, the above intersection is E^n by convention.) []

[P3.4.13] Property: (This is true of convex sets in general.) Let $U \subset E^n$ and put d=dimension of affine(U). Then

for each point $\underline{u} \in \text{convex}(U)$ there exist $d+1$ points $\underline{u}^i \in U$ such that $\underline{u} \in \text{convex}\{\underline{u}^1,\ldots,\underline{u}^{d+1}\}$. []

3.5 PROOF OF THEOREM

3.5.1 The Sets W And $\tilde{W}$

Consider the R-dimensional space E^R in which the vector $\underline{u}(\underline{x})$ lies, and consider any constraints vector $\underline{c}$ in this space. Following [L4] we define the Primal Functional

$$G(\underline{c}) \triangleq \max_{\underline{x} \in S_+} J(\underline{x}) \quad \text{subject to } \underline{u}(\underline{x}) \leq \underline{c}$$

and define $W \subset E^R$ as

$$W \triangleq \{ \underline{c} \mid \underline{c} \geq 0 \text{ and } G(\underline{c}) = J^* \}$$

so that W is the set of all constraints for which a feasible allocation exists for the original problem. Also let

$$\tilde{W} \triangleq \text{convex}(W) .$$

See Fig.3-2 for an illustration of W and $\tilde{W}$.

[D3.5.1] Definition: We shall say $\hat{\underline{c}}$ dominates $\underline{c}$ if $\hat{\underline{c}} \leq \underline{c}$, and we shall say $\hat{\underline{c}}$ is a non-dominated point (NDP) of $V \subset E^R$ if $\hat{\underline{c}} \in V$, and for all $\underline{c} \in V$ we have

$$\underline{c} \leq \hat{\underline{c}} \quad \Rightarrow \quad \underline{c} = \hat{\underline{c}}$$

(that is, $\hat{\underline{c}}$ is not dominated by any other point of V). See Fig.3-2 for examples. []

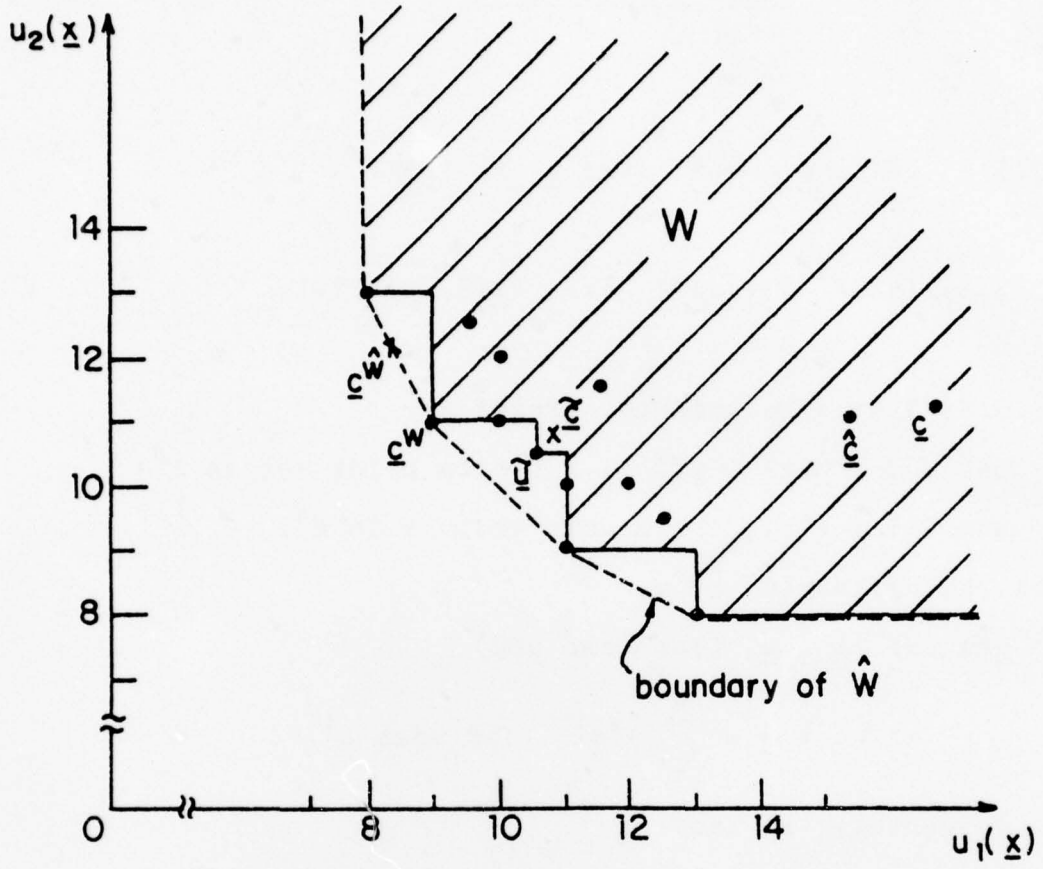


Fig.3-2: The sets W and $\hat{W}$ for the Example.

The set W consists of all constraints for which a feasible solution exists for the allocation example. $\hat{W}$ is the convex hull of W. It consists of all points to the right of, and above, the dotted line. $\underline{\hat{c}}$ dominates $\underline{c}$, and $\underline{c}^{W}$ is an example of a NDP of W, $\underline{c}^{\hat{W}}$ an NDP of $\hat{W}$. An example of duality gap is given by the problem with constraint $\underline{\tilde{c}}$ (see point x), for which there exists a solution $\underline{\tilde{u}} \leq \underline{\tilde{c}}$, but this solution does not lie on any support plane; in fact no point on any support plane can solve this problem.

We now make some observations on W and $\tilde{W}$, for later use.

[03.5.2] $\underline{\hat{c}} \in W$ and $\underline{\hat{c}}$ dominates $\underline{c}$ => $\underline{c} \in W$.

[03.5.3] ($\underline{x} \in S_+$, $J(\underline{x})=J^*$) <=> ($\underline{x} \in S$, $\underline{u}(\underline{x}) \in W$).

[03.5.4] $\tilde{W}$ is a polyhedron. Proof:

Let $U \hat{=} \{ \underline{u}(\underline{x}) \mid \underline{x} \in S \}$, a finite point set in E^R

and $V \hat{=} \{ \underline{e}^i \}$, the R unit vectors in E^R.

Then, by definition of W,

$$\underline{c} \in W \Rightarrow \underline{c} \geq \underline{u} \text{ (for some } \underline{u} \in U)$$

$$\Rightarrow \underline{c} = \underline{u} + \sum_{i=1}^{R} a^i \underline{e}^i \text{ (for some } a^i \geq 0)$$

and conversely, so that

$$W = U + \text{cone}(V) .$$

Now $\tilde{W} \hat{=} \text{convex}(W)$

$$= \text{convex}(U+\text{cone}(V)) = \text{convex}(U) + \text{cone}(V)$$

where the equivalence in the final step is easily established. Hence by [P3.4.2] $\tilde{W}$ is a polyhedron.

[03.5.5] $\underline{\hat{c}} \in \tilde{W}$ and $\underline{\hat{c}}$ dominates $\underline{c}$ => $\underline{c} \in \tilde{W}$ (compare [03.5.2]).

Proof: This follows from the final equation of [03.5.4].

3.5.2 On Representing The Boundary Planes Of $\tilde{W}$

[L3.5.6] Lemma: Let H be a boundary plane of $\tilde{W}$, which passes through $\underline{c}^* \in \tilde{W}$. Suppose H is represented as

[3.5.7] $$\{ \underline{c} \mid \underline{h}'\underline{c} = \underline{h}'\underline{c}^* \}$$

the sign of $\underline{h}$ being chosen such that

[3.5.8] $$\underline{h}'\underline{c}^* \leq \underline{h}'\underline{c} \quad \text{for all } \underline{c} \in \tilde{W} .$$

Then $\underline{h} \geq 0$ (componentwise). []

Proof: The proof is by contradiction. Suppose some components of $\underline{h}$ are negative. Let h_j be one such component. Consider $\bar{\underline{c}}$ such that

$$\bar{c}_j > c^*_j \quad \text{and} \quad \bar{c}_k = c^*_k \ (k \neq j).$$

Then

[3.5.9] $$\begin{aligned} \underline{h}'\bar{\underline{c}} &= \underline{h}'(\bar{\underline{c}} - \underline{c}^*) + \underline{h}'\underline{c}^* \\ &= h_j(\bar{c}_j - c^*_j) + \underline{h}'\underline{c}^* \\ &< \underline{h}'\underline{c}^* . \end{aligned}$$

But clearly $\bar{\underline{c}} \geq \underline{c}^*$ so that $\bar{\underline{c}} \in \tilde{W}$ (using [O3.5.5]) and thus from [3.5.8] $\underline{h}'\underline{c}^* \leq \underline{h}'\bar{\underline{c}}$, which contradicts [3.5.9]. []

[C3.5.10] Corollary: With definitions as in the previous Lemma, if $\underline{c}^*$ is dominated by $\bar{\underline{c}} \in \tilde{W}$, then $\bar{\underline{c}} \in H$. []

Proof: $\bar{\underline{c}} \in \tilde{W} \Rightarrow \underline{h}'\underline{c}^* \leq \underline{h}'\bar{\underline{c}}$ using [3.5.8].

$\underline{h} \geq 0$ and $\bar{\underline{c}} \leq \underline{c}^* \Rightarrow \underline{h}'\bar{\underline{c}} \leq \underline{h}'\underline{c}^*$. Hence $\underline{h}'\bar{\underline{c}} = \underline{h}'\underline{c}^*$. []

3.5.3 Characterization Of Non-dominated Faces

[D3.5.11] Definition: A proper face F of $\hat{W}$ such that some inner point* of F is an NDP of $\hat{W}$, is called a non-dominated face of $\hat{W}$. The set of all non-dominated faces of $\hat{W}$ will be denoted $\hat{N}$. (See Fig.3-3) []

The justification for naming an entire face, on the basis of one point, follows from the next Lemma.

[L3.5.12] Lemma: Let $\hat{N}$ be as above, and let $F \in \hat{N}$. Then

a. $\hat{N}$ includes all NDPs of $\hat{W}$.

b. All points of F are NDPs of $\hat{W}$.

c. F is bounded.

d. F is the convex hull of some extreme points of W.

e. If $\underline{c}^F \in F \cap W$, there exists $\underline{x} \in S$ with $\underline{u}(\underline{x}) = \underline{c}^F$.

[]

Note that (a) and (b) above show that the set of all points which are NDPs of $\hat{W}$ is equivalent to the set of all points on faces in $\hat{N}$.

---

*Note that from the definition of inner points [D3.4.10], a vertex is an inner point of itself.

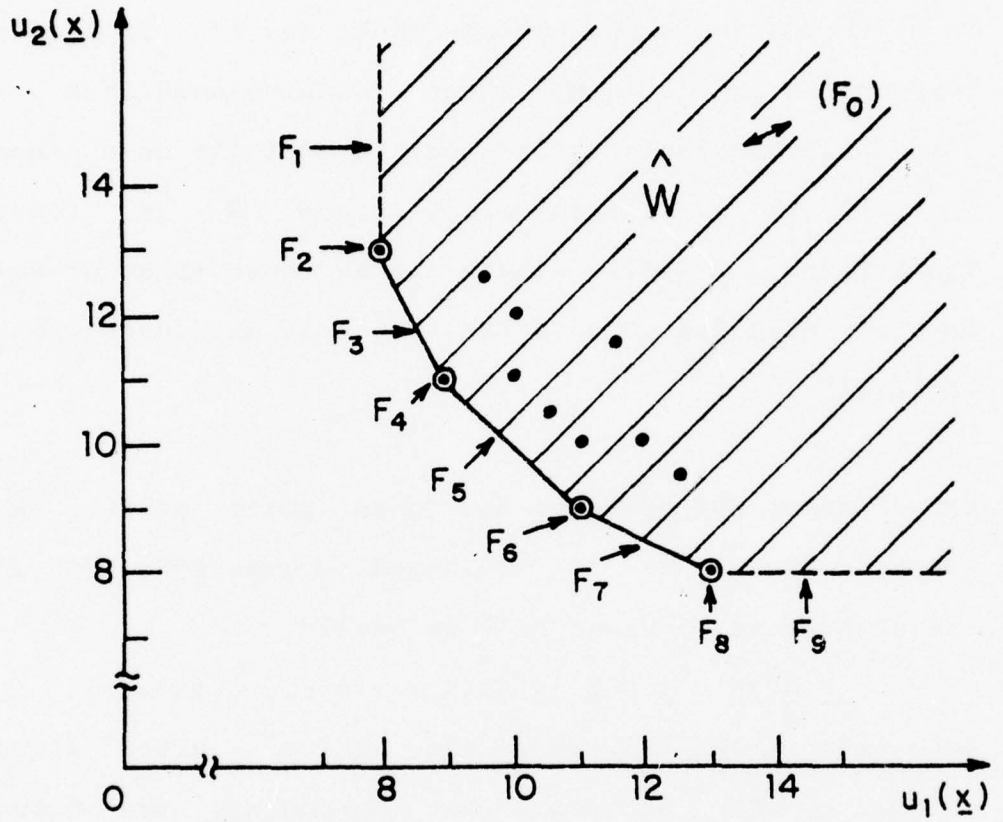


Fig.3-3: The 10 Faces of $\hat{W}$.

F_0 to F_9 are all the faces of $\hat{W}$.

F_2 to F_8 are non-dominated faces, members of $\mathbb{N}$.

The circled points ⊙ are examples of points in $F \cap W$, for some $F \in \mathbb{N}$ (see Lemma [L3.5.12]).

Proof:

a. Suppose $\underline{c}$ is an NDP of $\hat{W}$. Now $\underline{c}$ cannot be an inner point of $\hat{W}$, since it would be possible to find another point in $\hat{W}$ which would dominate it (see [P3.4.12]). Thus $\underline{c}$ must lie on a proper face, say F'. If $\underline{c}$ is an inner point of F', then $F' \in \mathbb{N}$ and $\underline{c}$ is included in $\mathbb{N}$. If $\underline{c}$ is a boundary point of F' then it must lie on a proper face F" of F'. Eventually, since $\hat{W}$ is finite dimensional, $\underline{c}$ will either be an inner point of some face, or will lie on a vertex, which is an inner point of itself. []

b. Let $\underline{c}^N$ be an NDP of $\hat{W}$ and an inner point of F. Now suppose that $\underline{c}^D \in F$ is dominated by some $\underline{\bar{c}} \in \hat{W}$. For any singular boundary plane of F we have**

$$\underline{h}'\underline{c}^N = \underline{h}'\underline{c}^D \leq \underline{h}'\underline{\bar{c}} \text{ (with } \underline{h} \text{ as in [L3.5.6]).}$$

But also $\underline{h} \geq 0$ [L3.5.6], and $\underline{\bar{c}} \leq \underline{c}^D$ with strict inequality for at least one component, say $\bar{c}_j < c_j^D$, implies $\underline{h}'(\underline{c}^D - \underline{\bar{c}}) = 0$ and $h_j = 0$. Thus if $\underline{e}^j$ is the unit vector in the j direction, $\underline{h}'\underline{e}^j = 0$. This means that $\underline{c}^N + \underline{e}^j$ lies on the intersection of all singular boundary planes of F so that [P3.4.12] there exists an $a > 0$ such that $\underline{c}^N - a\underline{e}^j \in F$. But this point dominates $\underline{c}^N$, which is a contradiction. Thus $\underline{c}^D$ must also be an

---

**We use this phrase frequently, to denote "a boundary plane of $\hat{W}$ which is a singular boundary plane of F ".

NDP of $\hat{W}$. []

c. F is a convex polyhedron [D3.4.4]. If it is not bounded, then [P3.4.9] there must exist $\underline{c}^F \in F$ and $\underline{c}$ such that

$$\underline{c}^F + b\underline{c} \in F \quad \text{for all } b \geq 0.$$

In addition, we must have some $c_i < 0$, or else $\underline{c}^F + b\underline{c}$ will be dominated by $\underline{c}^F$ (which cannot be, by (b) above). Choose b large enough so that

$$c_i^F + bc_i < 0$$

which is a contradiction, since W (and hence $\hat{W}$) consist of vectors with non-negative components. []

d. Since F is bounded, it must be the convex hull of its extreme points [P3.4.8]. These points, being faces of F, are also faces of $\hat{W}$ [D3.4.4] hence extreme points of $\hat{W}$. Let $\underline{c}$ be such a point. By definition of $\hat{W}$

$$\underline{c} = \text{convex}\{\underline{c}^1, \underline{c}^2, \ldots, \underline{c}^N\} \text{ for } \underline{c}^i \in W.$$

This implies that [D3.4.5] $\underline{c} = \underline{c}^i$ for all i, so that $\underline{c} \in W$. []

e. Since $\underline{c}^F \in W$, there exists $\underline{x} \in S_+$ with $\underline{u}(\underline{x}) \leq \underline{c}^F$ and $J(\underline{x}) = J^*$. But $\underline{c}^F$ is an NDP of $\hat{W}$, and $\underline{u}(\underline{x}) \in W \subset \hat{W}$, implies $\underline{u}(\underline{x}) = \underline{c}^F$. Also $J(\underline{x}) = J^* \Rightarrow \underline{x} \in S$. []

3.5.4 From Premises Of Theorem To Allocation On F

[L3.5.13] Lemma: (Delta-Domination Lemma)
For any $\underline{c}^F \in F \in \mathbb{F}$, $\underline{c}^F + \hat{a}\underline{\Delta}$ is dominated by some $\underline{c}^* \in W \cap F$, where $\hat{a}$ and $\underline{\Delta}$ are as in Theorem 3-I. (See Fig.3-4) []

Proof: This is quite involved and is left to the Appendix.

[L3.5.14] Lemma: If there exists $\underline{\bar{x}} \in S_+$ such that $J(\underline{\bar{x}}) = J^*$ and $\underline{u}(\underline{\bar{x}}) \leq \underline{\bar{c}} - \hat{a}\underline{\Delta}$, then there exists $\underline{x}^* \in S$ such that

a. $\underline{u}(\underline{x}^*) \leq \underline{\bar{c}}$

b. $\underline{u}(\underline{x}^*) \in F \cap W$, for some $F \in \mathbb{F}$.

[]

Proof: (See Fig.3-5) Since $J(\underline{\bar{x}}) = J^*$, $\underline{\bar{c}} - \hat{a}\underline{\Delta} \in W$. Now choose some $\underline{c}^F \in F$ (some $F \in \mathbb{F}$), which dominates $\underline{\bar{c}} - \hat{a}\underline{\Delta}$. (The reader can verify that such a choice is always possible. In particular if $\underline{\bar{c}} - \hat{a}\underline{\Delta}$ is itself an NDP, we can just choose $\underline{c}^F = \underline{\bar{c}} - \hat{a}\underline{\Delta}$.) Then from [L3.5.13], there exists a $\underline{c}^* \in F \cap W$ such that

$$\underline{c}^* \leq \underline{c}^F + \hat{a}\underline{\Delta}$$

$$\text{but also } \underline{c}^F \leq \underline{\bar{c}} - \hat{a}\underline{\Delta}$$

so that $\underline{c}^* \leq \underline{\bar{c}}$. But now since $\underline{c}^* \in F \cap W$, from [L3.5.12e] there exists some $\underline{x}^* \in S$ with $\underline{u}(\underline{x}^*) = \underline{c}^*$.

[]

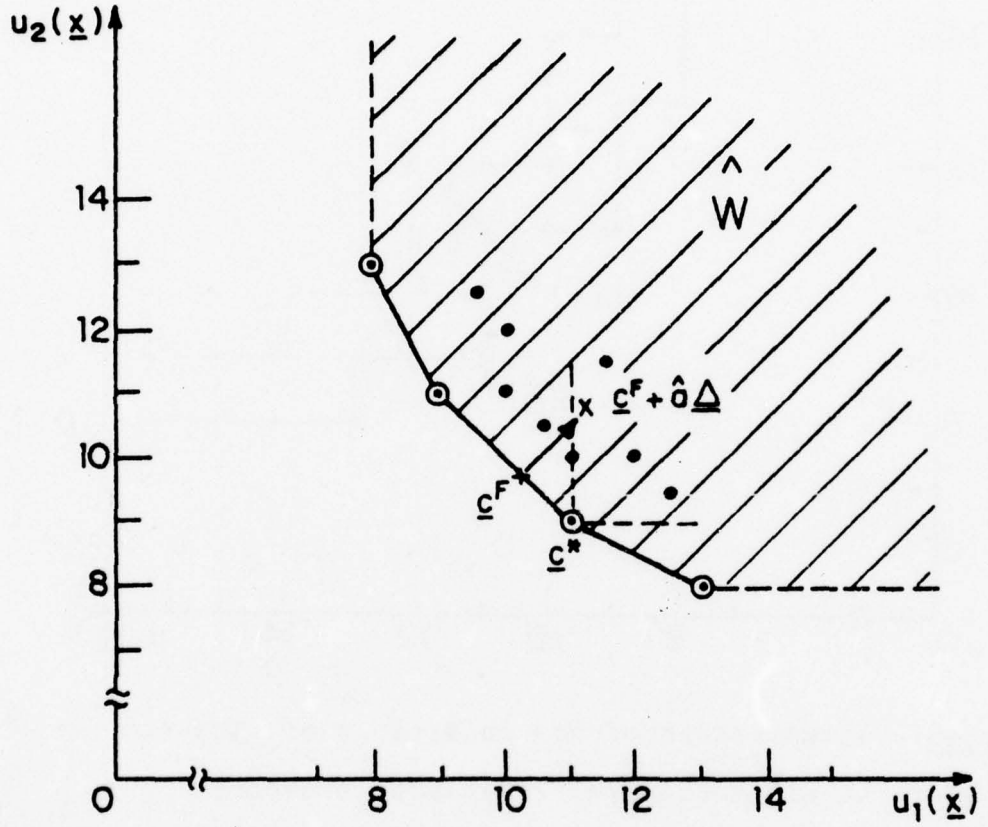


Fig.3-4: Illustration of the "Delta-Domination" Lemma. The vector $\hat{a}\underline{\Delta}$ equals [1,1]' for this example. It is shown by an arrow in the Figure.

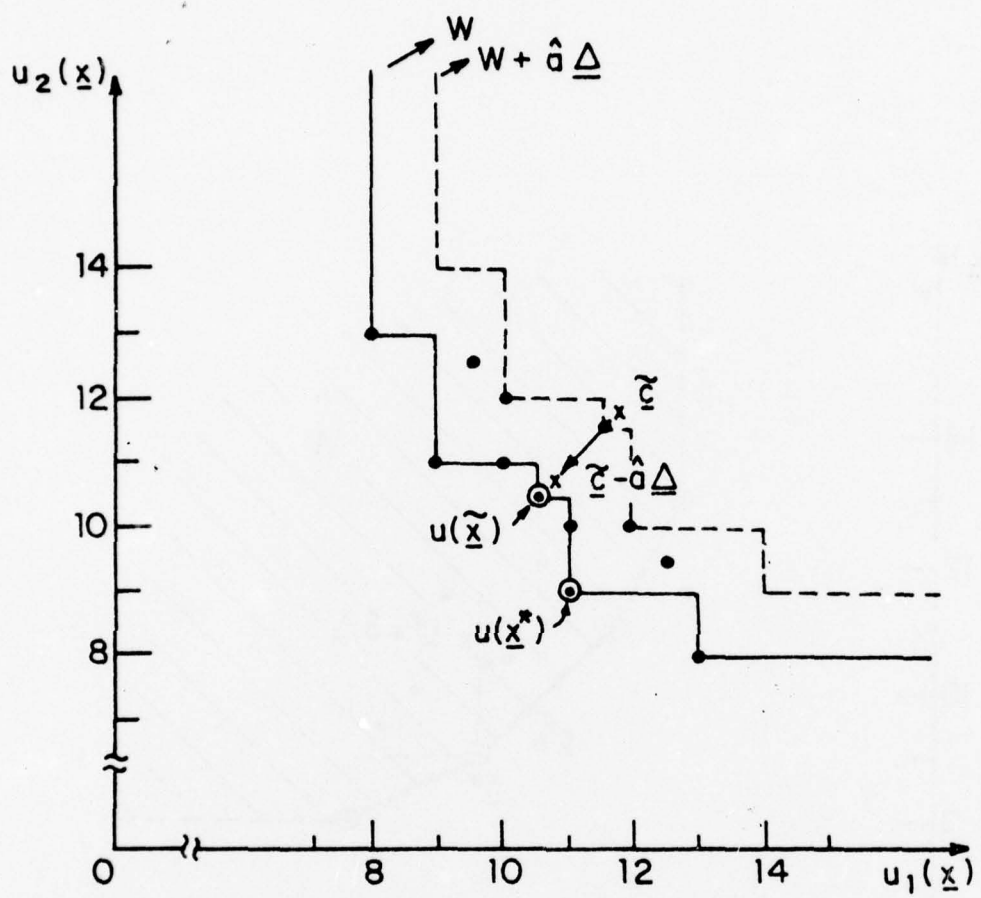


Fig.3-5: Illustration of how conditions of Theorem ensure existence of allocation on "reachable" boundary.

If there exists an allocation $\underline{x}$ with $\underline{u}(\underline{x}) \leq \underline{c} - \hat{a}\underline{\Delta}$, then $\underline{c}$ must lie in the set $W + \hat{a}\underline{\Delta}$ (dotted boundary). But then, from the Delta-domination Lemma, we know that there exists a $\underline{u}(\underline{x}^*)$ which dominates $\underline{c}$, and $\underline{u}(\underline{x}^*)$ lies on the "reachable" boundary.

3.5.5 Construction Of Suitable $\underline{\lambda}$

[L3.5.15] Lemma: If there exists $\underline{\hat{x}}$ as in the preceding Lemma, then there exists a $\underline{\lambda}^* \geq 0$, and an $\underline{\hat{x}}(\underline{\lambda}^*)$ as in [3.2.12], such that

$$J(\underline{\hat{x}}(\underline{\lambda}^*)) = J^*, \text{ and } \underline{u}(\underline{\hat{x}}(\underline{\lambda}^*)) \leq \underline{c} \quad []$$

Proof: Let $\underline{c}^*$ and $\underline{x}^*$ be as in the previous Lemma, and let $H, \underline{h}$ be as in [L3.5.6]. Now consider the R+1 dimensional space of Payoff versus resource Usage, with points $\begin{bmatrix} J \\ \underline{u} \end{bmatrix}$ such that $J=J(\underline{x})$, $\underline{u}=\underline{u}(\underline{x})$, for some $\underline{x} \in S_+$. Define the sets

$$\mathcal{J}_< \triangleq \{ \begin{bmatrix} J \\ \underline{u} \end{bmatrix} \mid J<J^* \}, \text{ and } \mathcal{J}_= \triangleq \{ \begin{bmatrix} J \\ \underline{u} \end{bmatrix} \mid J<J^* \}$$

Since the allocations are discrete points, $\mathcal{J}_<$ has a finite number of points and there exists a J' such that

$$J' = \max \{ J \mid \begin{bmatrix} J \\ \underline{u} \end{bmatrix} \in \mathcal{J}_< \}$$

Choose $b > \underline{h}'\underline{c}^*/(J^*-J')$. We will now show that

$$\underline{\lambda}^* = (1/b)\underline{h}$$

is a suitable value for $\underline{\lambda}$.

First, we note that (see [L3.5.6] and definition of J') $\underline{\lambda}^* \geq 0$.

Next, consider any $\begin{bmatrix} J \\ \underline{u} \end{bmatrix} \in \mathcal{J}_<$. Then

$$b > \underline{h}'\underline{c}^*/(J^*-J) \qquad \text{since } J \leq J'$$

$$> (\underline{h}'\underline{c}^*-\underline{h}'\underline{u})/(J^*-J) \qquad \text{since } \underline{h} \geq 0,\ \underline{u} \geq 0$$

$$\Rightarrow \quad J^*-J > (\underline{h}'\underline{c}^*-\underline{h}'\underline{u})/b$$

that is,

[3.5.16] $\quad J^*-\underline{\lambda}^{*\prime}\underline{c}^* > J-\underline{\lambda}^{*\prime}\underline{u}$.

And now, consider any $\begin{bmatrix} J \\ \underline{u} \end{bmatrix} \in \mathcal{J}_=$. Then

$$J^* - \underline{h}'\underline{c}^*/b = J - \underline{h}'\underline{c}^*/b \quad \text{since } J=J^*$$
$$\geq J - \underline{h}'\underline{u}/b \quad (\underline{u}\in W, \text{ and using [L3.5.6]})$$

so that

[3.5.17] $\quad J^*-\underline{\lambda}^{*\prime}\underline{c}^* \geq J-\underline{\lambda}^{*\prime}\underline{u}$.

Finally, we consider that $\hat{\underline{x}}(\underline{\lambda}^*)$ maximizes $J(\underline{x})-\underline{\lambda}^{*\prime}\underline{u}(\underline{x})$ over $\underline{x}\in S_+$. From [3.5.15] and [3.5.16] we see that $\hat{\underline{x}}(\underline{\lambda}^*)=\underline{x}^*$ ($\underline{x}^*$ as in [L4.5.14]) satisfies this criterion. Furthermore, from [L3.5.14] it follows that $\underline{c}^*\leq \bar{\underline{c}}$, so that we have $\underline{u}(\underline{x}^*)=\underline{c}^*\leq \bar{\underline{c}}$, and $J(\underline{x}^*)=J^*$. We have thus shown the existence of a suitable $\underline{\lambda}^*$ and $\hat{\underline{x}}(\underline{\lambda}^*)$. []

This also concludes the proof of Theorem 3-I. []

3.6 EXISTENCE OF STRICTLY POSITIVE $\underline{\lambda}$

Under the stated conditions, Theorem 3-I showed the existence of a suitable $\underline{\lambda} \geq 0$. From the proof of [L3.5.15] we see that, since we could choose <u>any</u> support plane for $\hat{W}$ through $\underline{c}^*$, and a range of values of b, the $\underline{\lambda}$ is not unique -- there is a range of possible values. The next Theorem shows that in this range of values there also lies a $\underline{\lambda}$ which is strictly positive. (This result will be useful in Chapter 4, for our iteration algorithm.)

[T3.6.1] Theorem 3-II:

Under the conditions of Theorem 3-I, there also exists a $\underline{\lambda}^*>0$ such that $\underline{x}(\underline{\lambda}^*)$ solves (AP). []

Proof: Let $\underline{c}^* \in F$, for some $F \in \mathcal{F}$, be as in the proof of [L3.5.14]. Since F is a face of $\hat{W}$, from [D3.4.4] it can be written

[3.6.2] $$F = \hat{W} \cap H^1 \cap H^2 \cap \ldots \cap H^N$$

where the H^i are boundary planes of $\hat{W}$. Since $\underline{c}^*$ lies on each H^i, we can write each H^i in the form

$$\{ \underline{c} \mid \underline{h}^i{}'\underline{c} = \underline{h}^i{}'\underline{c}^* \}$$

and by choosing the sign of each $\underline{h}^i$ such that

[3.6.3] $$\underline{h}^i{}'\underline{c}^* \leq \underline{h}^i{}'\underline{c} \quad \text{for all } \underline{c} \in \hat{W},$$

we shall have $\underline{h}^i \geq 0$ (from [L3.5.6]). Now let

$$\underline{n} \triangleq \sum_{i=1}^{N} \underline{h}^i .$$

Then $\underline{n}'\underline{c}^* \leq \underline{n}'\underline{c}$ for all $\underline{c} \in \hat{W}$ from [3.6.3].

From above we know that $\underline{n} \geq 0$. We will now prove, by contradiction, that $\underline{n}>0$ (strict). Suppose $n_j=0$. This implies $h^i_j=0$ for all i. Let $\hat{\underline{c}} \triangleq \underline{c}^* + a\underline{e}^j$, where $\underline{e}^j$ is the unit vector in the j direction, and $a>0$. Then $\hat{\underline{c}}$ is dominated by $\underline{c}^*$ so that, from [03.5.5], $\hat{\underline{c}} \in \hat{W}$. Also, for all $i=1,\ldots,N$

$$\underline{h}^i{}'\hat{\underline{c}} = \underline{h}^i{}'\underline{c}^* + a\underline{h}^i{}'\underline{e}^j = \underline{h}^i{}'\underline{c}^*$$

so that $\hat{\underline{c}} \in H^i$. Hence by [3.6.2], $\hat{\underline{c}} \in F$. But $\hat{\underline{c}}$ is dominated by $\underline{c}^*$, and this contradicts the property [L4.5.12b] of non-dominated faces. Thus $n_j>0$.

The remainder of the proof follows [L3.5.15] exactly. []

[C3.6.4] Corollary: If there exists a $\underline{\lambda} \geq 0$ such that $\underline{\hat{x}}(\underline{\lambda})$ solves (AP), then there exists a $\underline{\lambda}^* > 0$ (strictly) such that $\underline{\hat{x}}(\underline{\lambda}^*) = \underline{\hat{x}}(\underline{\lambda})$, so that $\underline{\hat{x}}(\underline{\lambda}^*)$ solves (AP). []

Proof: In the proof of Theorem 3-II above, simply take $\underline{c}^* = \underline{u}(\underline{\hat{x}}(\underline{\lambda}))$. []

[R3.6.5] Remark: The â$\underline{\Delta}$ condition is not required for this corollary. []

3.7 DISCUSSION

Everett [E1] pointed out that, even under general conditions on the $\underline{u}^i(.)$ and S^i, some statements could be made regarding the properties of an allocation $\underline{\hat{x}}(\underline{\lambda})$, for any $\underline{\lambda} \geq 0$. He did not, however, deal with the existence of optimal multipliers. Our results greatly extend the applicability of the Lagrange Multiplier technique, by giving simple conditions for the existence of optimal multipliers, under general conditions on the $\underline{u}^i(.)$ and S^i. We also emphasize that in practice, our conditions are likely to hold for large systems. This justifies the use of the decentralized method of finding feasible solutions to large allocation problems.

APPENDIX 3A

THE DELTA-DOMINATION LEMMA

For the convenience of the reader, the Delta-Domination Lemma [L3.5.13] is re-stated here:

For any $\underline{c}^F \in F \in \mathcal{F}$, $\underline{c}^F + \hat{a}\underline{\Delta}$ is dominated by some $\underline{c}^* \in W \cap F$. (Here $\hat{a}$ and $\underline{\Delta}$ are as in Theorem 3-I.) []

Outline of Proof of the Lemma

The proof proceeds in five stages:

1. We characterize the set of all points in $W \cap F$, in terms of the resource usage vectors of individual activities.

2. We represent these individual usage vectors as basis vectors in a different space, and show how a certain polyhedron, P, represents F in this space.

3. We show that a certain subset of the faces of P maps onto all of F.

4. We derive a simple representation of any such face of P.

5. We take a point $\underline{c}^F$ on F, and choose a certain point on a suitable face of P. This point will map onto $\underline{c}^* \in W \cap F$ such that $\underline{c}^* \leq \underline{c}^F + \hat{a}\underline{\Delta}$.

3A.1 SET OF POINTS IN $F \cap W$

Let $\underline{c}_j$ be a point of $F \cap W$. (By [L3.5.12d] $F \cap W$ cannot be vacuous.) Then, by [L3.5.12e] there exists an allocation $\underline{x}_j$ with $\underline{u}(\underline{x}_j) = \underline{c}_j$. In this allocation, let

$$\underline{u}_j^i \triangleq \text{usage vector of activity } i$$

so that

$$\underline{c}_j = \sum_{i=1}^{I} \underline{u}_j^i \ .$$

Now, for all points $\underline{c}_j \in W \cap F$, let $\underline{u}_j^i$ be defined as above, and define the set

$$U^i \triangleq \bigcup_j \{\underline{u}_j^i\} \ .$$

Loosely speaking, U^i is the "set of usage vectors of the i^{th} Activity, for points lying on $F \cap W$".

Our first step consists of showing that <u>all</u> combinations of these $\underline{u}_j^i$ according to the manner

[3A.1.1] $$\hat{\underline{c}} = \sum_{i=1}^{I} \hat{\underline{u}}^i \qquad \text{for} \quad \hat{\underline{u}}^i \in U^i$$

will generate points $\hat{\underline{c}}$ also lying in $F \cap W$. In other words, we show that the set of <u>all</u> combinations $\hat{\underline{c}}$ in [3A.1.1], is

exactly the set of all points of $F \cap W$.

Suppose $\underline{c}_+$ and $\underline{c}_o$ are both in $F \cap W$. Let

$$\underline{c}_+ \hat{=} \sum_{i=1}^{I} \underline{u}_+^i \quad \text{for} \quad \underline{u}_+^i \in U^i$$

$$\underline{c}_o \hat{=} \sum_{i=1}^{I} \underline{u}_o^i \quad \text{for} \quad \underline{u}_o^i \in U^i$$

and let H be any singular boundary plane of F (see [D3.4.11]) with equation

[3A.1.2] $\quad \underline{h}'\underline{c}_+ \leq \underline{h}'\underline{c} \quad$ for all $\underline{c} \in W$

so that, by definition,

[3A.1.3] $\quad \underline{h}'\underline{c}_+ = \underline{h}'\underline{c}_o$.

We will show that $\underline{h}'(\underline{u}_o^i - \underline{u}_+^i) = 0$ for all i. Suppose that $\underline{h}'(\underline{u}_o^k - \underline{u}_+^k) < 0$ for some k ($1 \leq k \leq I$). Consider the allocation

$$\underline{c} \hat{=} \sum_{i=1}^{I} \underline{u}^i \quad \text{where} \quad \begin{cases} \underline{u}^i = \underline{u}_+^i & (i \neq k) \\ \underline{u}^k = \underline{u}_o^k \end{cases}$$

Since $\underline{c} \in W$, we must have

$$\begin{aligned} \underline{h}'\underline{c}_+ &\leq \underline{h}'\underline{c} \\ &= \underline{h}'(\underline{c}_+ + \underline{u}_o^k - \underline{u}_+^k) \\ &< \underline{h}'\underline{c}_+ \end{aligned}$$

which is a contradiction. Thus we have shown that

$$\underline{h}'(\underline{u}_o^i - \underline{u}_+^i) \geq 0 \quad \text{for all i .}$$

But also, from [3A.1.3],

$$\underline{h}' \sum_{i=1}^{I} (\underline{u}_o^i - \underline{u}_+^i) = 0 .$$

Hence

[3A.1.4] $\quad \underline{h}'(\underline{u}_o^i - \underline{u}_+^i) = 0 \quad$ for all i.

Clearly, by definition of U^i, and the generality of $\underline{c}_+$, $\underline{c}_o$ above, we can generalize [3A.1.4] to

$$\underline{h}'(\underline{u}_o^i - \underline{u}_+^i) = 0 \quad \text{for all } \underline{u}_o^i, \underline{u}_+^i \in U^i; \text{ for all } i.$$

It is trivial to show now, that for any $\underline{\hat{c}}$ as in [3A.1.1], we will have

$$\underline{h}'\underline{\hat{c}} = \underline{h}'\underline{c}_o \qquad (\underline{c}_o \text{ as in [3A.1.3]})$$

so that $\underline{\hat{c}}$ also lies on H. Also $\underline{\hat{c}} \in W$. Hence $\underline{\hat{c}} \in H \cap \hat{W}$.

Since H is any boundary plane containing F, it follows that $\underline{\hat{c}}$ lies on the intersection of all such planes with $\hat{W}$. But this is precisely the definition of F [D3.4.4]. Thus $\underline{\hat{c}} \in F \cap W$.

3A.2 REPRESENTATION OF F IN A HIGHER-DIMENSIONAL SPACE

In the following, for ease of notation, we shall use doubly-indexed unit vectors $\underline{v}_j^i$, by which we mean

$$\underline{v}_j^i = \underline{v}_m^k \iff i=k,\ j=m$$

$$\text{and } \underline{v}_j^i \neq \underline{v}_m^k \iff \underline{v}_j^i \perp \underline{v}_m^k .$$

Now let the N^i elements of U^i be ordered in some fashion, say

$$U^i = \{\underline{u}_1^i, \underline{u}_2^i, \ldots, \underline{u}_{N^i}^i\} .$$

We shall, as shown below, associate these elements with unit vectors in the set

$$V^i \triangleq \{\underline{v}_1^i, \underline{v}_2^i, \ldots, \underline{v}_{N^i}^i\} .$$

Let $V \triangleq \bigcup_{i=1}^{I} V^i$ and $N \triangleq \sum_{i=1}^{I} N^i$ = dimension of span(V).

Define the linear map T(.) from span(V) to $\text{span}(U^1+U^2+\ldots+U^I)$ by

$$T(\underline{v}_k^i) \triangleq \underline{u}_k^i \ .$$

Since the $\underline{v}_k^i$ are basis vectors for span(V), the above equation defines T(.) for all points in span(V) [N1].

We now consider those points in span(V) which can be written

[3A.2.1] $$\underline{v} = \sum_{i=1}^{I} \underline{v}^i \quad \text{for } \underline{v}^i \in V^i \ .$$

Clearly, $T(\underline{v}) = \hat{\underline{c}}$, for some $\hat{\underline{c}}$ as in [3A.1.1], so that these points correspond one-to-one with all points in $F \cap W$. Let P^o be the set of all such $\underline{v}$, and P its convex hull (so that by [P3.4.2] P is a Polyhedron).

Since F is the convex hull of its extreme points, all of which lie in $F \cap W$ [L3.5.12d], the map T transforms points in P to points on F. (The map is obviously many-to-one, in general.)

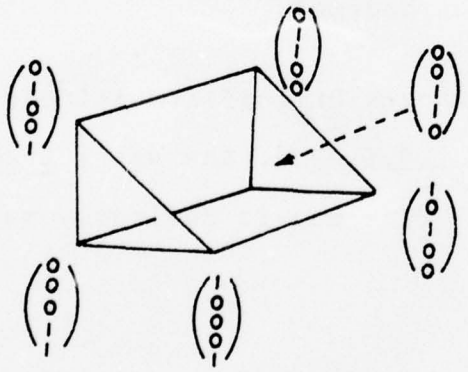
Remark: P is not a hypercube, in general. For example, in 5 dimensions, if we take

V^1 = {[1,0,0,0,0]', [0,1,0,0,0]', [0,0,0,1,0]'}

V^2 = {[0,0,1,0,0]', [0,0,0,0,1]'}

then the 3-D projection of the object P is the prism in Fig.3A-1. (We are projecting both [0,0,0,1,0]' and [0,0,0,0,1]' into [0,0,0]'.)

Fig.3A-1: The Object P



3A.3 ONLY NEED TO CONSIDER CERTAIN FACES OF P

Let n=dimension of F. In this step we show that the map of just the r-dimensional faces of P (all $r \leq n$), is sufficient to cover F. This will mean that, in later steps, we need only consider such faces. Formally, we have:

[3A.3.1] Lemma: Let n=dimension of F. For any point $\underline{c} \in F$, there exists a point $\underline{v} \in G$ (an r-dimensional face of P, $r \leq n$) such that $\underline{c} = T(\underline{v})$. []

Proof: (Reminder: P is N-dimensional, F is n-dimensional.) If $N \leq n$, there is nothing to be shown. In practice of course we expect $N >> n$. Now if $N > n$, let Y be the N-n dimensional null-space of T, and let $\underline{c}=T(\underline{w})$ for some $\underline{W} \in P$. The linear variety $\underline{w}+Y$ can be written as the intersection of n Hyperplanes in the space of span(V):

[3A.3.2] $$\underline{w}+Y = \bigcap_{i=1}^{n} H_i \ .$$

Consider $(\underline{w}+Y) \cap P$. Suppose this intersection is contained entirely within a face of P, defined by the intersection of s boundary planes

[3A.3.3] $$\bigcap_{i=n+1}^{n+s} H_i$$

with P. (This statement is quite general, since if s=0 then this intersection is, by convention, the whole space, and the face is just P itself.)

Combining [3A.3.2] and [3A.3.3] we have

$$(\underline{w}+Y)\cap P \subset \bigcap_{i=1}^{n+s} H_i \ .$$

Since not all of the s boundary planes are necessarily different from the first n planes, we can rewrite the above as

[3A.3.4] $$(\underline{w}+Y)\cap P \subset \bigcap_{i=1}^{r+s} H_i \quad \text{for some } r\leq n \ .$$

Now $(\underline{w}+Y)$ is itself a polyhedron. Consider a vertex $\underline{v}$ of this polyhedron, which by [3A.3.4] must be obtained by the intersection with N-(r+s) more boundary planes of P, none of which is the same as any H_i in [3A.3.4] above. Now $\underline{v}$ lies on these N-(r+s) boundary planes of P, as well as on the s boundary planes of P in [3A.3.3], so that it lies on the intersection of N-(r+s)+s = N-r different boundary planes with P. Thus $\underline{v}$ lies on an r-dimensional face of P. But also $\underline{v}$ lies in $\underline{w}+Y$, so that $T(\underline{v})=\underline{c}$ as desired. []

3A.4 REPRESENTATION OF r-DIMENSIONAL FACES OF P

Let G be an r ($\leq n$) dimensional face of P, as in the previous section. Below, we derive a simple representation for any point on G. Let

[3A.4.1] $$\underline{v}_o = \sum_{i=1}^{I} \underline{v}_o^i \quad \text{for } \underline{v}_o^i \in V^i \text{ (similar to [3A.2.1])}$$

and let $Z^{k1},\ldots,Z^{kt}$ be such that

$$Z^{kj} \subset V^{kj}$$

$$\text{with } \underline{v}_o^{kj} \notin Z^{kj}$$

and cardinality of $Z^{kj} = r^{kj} \geq 1$, with $\sum_{j=1}^{t} r^{kj} = r$.

Thus in general,

[3A.4.2] $t \leq r$

since some $Z^{k}j$ may have more than one element.

Then the set of all points

[3A.4.3] $\underline{v}_{+} = \underline{v}_{o} + \sum_{j=1}^{t}(\underline{v}_{+}^{k}j - \underline{v}_{o}^{k}j)$ for $\underline{v}_{+}^{k}j \in Z^{k}j \cup \{\underline{v}_{o}^{k}j\}$

is the set of extreme points of an r-dimensional face of P. In fact, extreme points of <u>any</u> r-dimensional face of P can be thus represented, by appropriate choice of $\underline{v}_{o}$ and $Z^{k}j$. (This can be proved in a manner exactly paralleling that in sec.3A.1).

Now all points on G are the convex hull of G's extreme points. Using [P3.4.13], for any point $\underline{v} \in G$ there exist r+1 points $\underline{v}_{+}^{1}, \underline{v}_{+}^{2}, \ldots, \underline{v}_{+}^{r+1} \in G$ with

$$\underline{v}_{+}^{m} = \underline{v}_{o} + \sum_{j=1}^{t}(\underline{v}_{+}^{k}j;m - \underline{v}_{o}^{k}j) \qquad \text{(see [3A.4.3])}$$

such that $\underline{v} = \sum_{m=1}^{r+1} b^{m}\underline{v}_{+}^{m}$ for some $b^{m} \geq 0$, $\sum_{m=1}^{r+1} b^{m} = 1$

which implies

[3A.4.4] $$\underline{v} = \underline{v}_{o} + \sum_{m=1}^{r+1} b^{m} \sum_{j=1}^{t}(\underline{v}_{+}^{k}j;m - \underline{v}_{o}^{k}j) \ .$$

This is the representation that we desire.

3A.5 DERIVATION OF BOUND $\hat{a}\underline{\Delta}$

Let $\underline{c}^F$ be any point on F. From [3A.3.1] we know that $\underline{c}^F = T(\underline{v})$ where $\underline{v}$ is as in [3A.4.4]. Thus

$$\underline{c}^F = T(\underline{v}_o) + \sum_{m=1}^{r+1} b^m \sum_{j=1}^{t} T(\underline{v}_+^{kj;m} - \underline{v}_o^{kj}) \quad \text{using [3A.4.4]}$$

Define $\underline{u}_o^i \triangleq T(\underline{v}_o^i)$ so that $\underline{u}_o^i \in U^i$

$$U_+^{kj} \triangleq T(\{\underline{v}_o^{kj}\} \cup Z^{kj}) \triangleq \{\underline{u}_o^{kj}, \underline{u}_1^{kj}, \ldots, \underline{u}_{r_{kj}}^{kj}\} \text{ (say)}$$

so that $U_+^{kj} \subset U^{kj}$. Let $\underline{u}^{kj;m} \in U_+^{kj}$.

Then we can write

$$\underline{c}^F = \sum_{i=1}^{I} \underline{u}_o^i + \sum_{m=1}^{r+1} b^m \sum_{j=1}^{t} (\underline{u}^{kj;m} - \underline{u}_o^{kj})$$

$$= \sum_{i=1}^{I} \underline{u}_o^i + \sum_{j=1}^{t} \sum_{p=0}^{r^{kj}} g_p^{kj} (\underline{u}_p^{kj} - \underline{u}_o^{kj})$$

where $g_p^{kj} \geq 0$ and $\sum_{p=0}^{r^{kj}} g_p^{kj} = 1$ for each j (j=1,...,t) .

For each k_j (j=1,...,t) choose the constants h_p^{kj} as follows:

1. For p=0 to r^{kj}, set $h_p^{kj}=0$.
2. For p=1 to r^{kj}, if $g_p^{kj}>0.5$, set $h_p^{kj}=1$.
3. If all h_p^{kj} (p=1 to r^{kj}) are zero, then set $h_0^{kj}=1$.

[3A.5.1] Remark: From the properties of g_p^{kj}, we see that for each k_j, the above procedure sets exactly one $h_p^{kj}=1$ (p=0,...,r^{kj}) and the rest are set to zero. □

Define $\underline{c}^* = \sum_{i=1}^{I} \underline{u}_o^i + \sum_{j=1}^{t} \sum_{p=0}^{r^{kj}} n_p^{kj}(\underline{u}_p^{kj} - \underline{u}_o^{kj})$.

From [3A.5.1] we see that

$$\underline{c}^* = \sum_{i=1}^{I} \underline{u}^i \text{ for some } \underline{u}^i \in U^i,$$

so that $\underline{c}^* \in F \cap W$.

Now $$\underline{c}^* - \underline{c}^F = \sum_{j=1}^{t} \sum_{p=0}^{r^{kj}} (g_p^{kj} - h_p^{kj})(\underline{u}_p^{kj} - \underline{u}_o^{kj})$$

$$\leq 0.5r\underline{\Delta}$$

$$\leq 0.5n\underline{\Delta} \qquad \text{(see [3A.3.1])}$$

$$\leq 0.5(R-1)\underline{\Delta}$$

where the final inequality follows from the fact that F is a proper face of the R-dimensional polyhedron $\bar{W}$. Thus we have

$$\underline{c}^* \leq \underline{c}^F + 0.5(R-1)\underline{\Delta} \qquad [\,]$$

CHAPTER 4

THE "SALA" (STORAGE ALLOCATION ALGORITHM) TECHNIQUE

4.1 OVERVIEW OF CHAPTER

In Chapter 2 we described the Resource Management problem in a very large system, and showed why the decentralized approach may be a good solution technique for this problem. In order for this approach to be usable we must overcome the problems described in sec.2.7 which are briefly (i) possibility of duality gaps, (ii) convergence of algorithms requires stringent conditions on the functions, and (iii) slow convergence of such algorithms.

With respect to the first of these problems, in Chapter 3 we gave simple conditions which guaranteed the existence of optimal multipliers for the Artificial problem, under very general conditions on the functions. Resolving the other two of the above problems then, shall be the main contribution of this Chapter. We shall develop an iteration algorithm with proveable convergence properties, and a quadratic convergence rate. The reader should notice how (both in this Chapter and in Chapter 3) we are able to completely remove restrictions on the form of the individual

resource usage functions, replacing them instead by some restrictions on their magnitudes. We believe that this is an important philosophy for analysis of large-scale systems in general, since the former requirements (convexity/ continuity/ linearity) are seldom met in practice, whereas the latter are easily verified.

A minor, but nevertheless interesting, contribution of this Chapter is the Selection Algorithm (sec.4.4). This is a simple and intuitively appealing method of solving certain inequality problems, yet its solution, it turns out, possesses useful minimum norm properties.

4.2 THE SALA APPROACH

Our approach, called SALA (for Storage Allocation Algorithm, a name which refers to its original objective at the warehouse) will be as follows: We begin by observing that the Individual Problem (IP) [2.6.10] can be made still easier. Then we look for an iteration scheme to find a suitable $\underline{\lambda}^*$ (suitable in the sense of sec.2.7). To do this we first make some simplifying assumptions. The resulting model will be analyzed; we shall propose an algorithm and study its properties. This will provide us with insight as to how to extend our algorithm to the more realistic case. Finally we shall prove convergence results for this latter case.

4.2.1 Preview Of Iteration Scheme

We remind the reader of our objective which (as described in sec.2.7) is to find a $\underline{\lambda}^*$ such that $\underline{x}(\underline{\lambda}^*)$ and $\underline{u}(\underline{\lambda}^*)$ are optimal for the Artificial Problem*.

Our iteration scheme is motivated partly by the Arrow-Hurwicz scheme described in Chapter 2 (sec.2.7), and partly by our existence theorem (Chapter 3). We choose a starting value of $\underline{\lambda}$, say $\underline{\lambda}^o$, and then follow the scheme

[4.2.1] $$\underline{x}^{k+1} = \arg \min_{\underline{x} \in X^*} L^*(\underline{x}, \underline{\lambda}^k)$$

[4.2.2] $$\underline{\lambda}^{k+1} = \underline{\lambda}^k + \underline{\Delta\lambda}^k$$

until we find an optimal $\underline{\lambda}$. The reader should compare the above scheme with [2.7.1] and [2.7.2]. Firstly, our scheme replaces "max L" for $\underline{x} \in X$ by "min L*" for $\underline{x} \in X^*$, where (see next section) L* and X* are such that they further simplify the solution to [4.2.1] as compared with [2.7.1]. Secondly, we have a different method of updating $\underline{\lambda}$, using the Selection Algorithm (see sec.4.4), which will lead to a quadratic convergence rate of the scheme [4.2.1] and [4.2.2].

---

*The Saddle point condition was stated during our review of known theory. From the discussion at the end of sec.3.2.3, and the Theorem in sec.3.3, we see that it is not required for our solution.

4.2.2 "Min-Cost" Allocation For Each Item

Let us consider the Individual Problem (IP) [2.6.10] once again. It is easy to see that if there exists a $\underline{\lambda}$ such that $\underline{x}(\underline{\lambda})$ achieves the maximum value of (AP) [2.6.1], then (IP) can be replaced by the following Minimum Cost Allocation Problem*

[4.2.3] (MCA): $$\min_{\{\underline{x}^i \mid \underline{e}'\underline{x}^i = Q^i\}} [\underline{\lambda}'\underline{u}^i(\underline{d}^i, \underline{x}^i)]$$

This says that for a given set of "costs" $\underline{\lambda}$, the i^{th} item must find that allocation $\underline{x}^i$ (of all its quantity Q^i) which minimizes its total resource usage cost. Thus, assuming the existence of an optimal $\underline{\lambda}$ (see next section), and summing (MCA) over all i, we have replaced [2.7.1] by [4.2.1], provided we define

[4.2.4] $$L^*(\underline{x}, \underline{\lambda}) = \underline{\lambda}'\underline{u}(\underline{x}),$$

and X^* as the set of $\underline{x}$ such that each $\underline{x}^i$ in $\underline{x}$ satisfies the equality constraint in (MCA). The iteration scheme using L^* in [4.2.1] then has the following interpretation: we hold the objective function $J(\underline{x})$ at its maximum value ($\sum_{i=1}^{I} Q^i$) and try to bring the total resource usage $\underline{u}(\underline{x})$ into the feasible region ($\leq \underline{c}$).

---

*Reminder: $\underline{u}^i(\underline{d}^i, \underline{x}^i)$ = vector of resource usages by item i, whose data vector is $\underline{d}^i$, when its allocation is $\underline{x}^i$.
And $\underline{e} = [1, 1, \ldots, 1]'$.

The equality constraint in (MCA), and the fact that $\underline{x}^i$ is now an S-dimensional vector, make (MCA) a relatively simple problem. To illustrate this we note that in the special case where the $\underline{u}^i(\underline{d}^i,.)$ are linear functions of $\underline{x}^i$, the solution to (MCA) is simply to allocate all the quantity to one storage area.

4.2.3 Existence Of Optimal $\underline{\lambda}$

The conditions for the existence of an optimal $\underline{\lambda}$ were given in Theorem 3-I. Our proof there depended on the assumption that the strategy set for each $\underline{x}^i$ was discrete. Let us simply restrict each $\underline{x}^i$ to a discrete set. (This means each $\underline{x}^i$ may assume only a finite number of values. For example, we could restrict all the quantity of one part to be in one storage-type only, giving a strategy set of size S^i = number of storage-types. In theory, we could let this discrete set be as finely divided as required, so this should not be a major restriction.) This restriction actually has two advantages: (i) we can apply Theorem 3-I, and (ii) it makes the solution of (MCA) simpler.

Now, the conditions in Theorem 3-I, stated informally, are "if we reduced the limits vector $\underline{c}$ by the maximum resource usage of about 12 parts, the problem would still be feasible". Since we are dealing with 50,000 parts, this condition is highly likely to hold. Thus we are justified in proceeding as if an optimal $\underline{\lambda}$ exists.

AD-A064 780 HARVARD UNIV CAMBRIDGE MA DIV OF APPLIED SCIENCES F/G 5/1
RESOURCE MANAGEMENT IN LARGE SYSTEMS.(U)
DEC 78 R SURI
UNCLASSIFIED TR-671 N00014-75-C-0648
NL

2 OF 3
AD
A064780

4.3 DISCUSSION OF MAIN ASSUMPTIONS

As mentioned in the overview, we shall first make some simplifying assumptions, and analyze the idealized model. Let W be a bounded subset of the $\underline{\lambda}$ space, which will be delimited later. The starting point for our analysis is

[A4.3.1] Assumption: For all $\underline{\lambda} \in W$, the function $\underline{u}(\underline{\lambda})$ is continuous and Frechet differentiable. []

This assumption will be relaxed in sec.4.6. Note that we do not, however, make any convexity (or concavity) assumptions as in Arrow et.al. [A1] or Zangwill [Z1]. In view of [A4.3.1] we will define the Jacobian of $\underline{u}(\underline{\lambda})$ at any $\underline{\lambda}'' \in W$ by the matrix $A(\underline{\lambda}'')$, that is

[D4.3.2] Definition $$A_{ij}(\underline{\lambda}'') \triangleq \frac{\partial u_i}{\partial \lambda_j}\bigg|_{\underline{\lambda}=\underline{\lambda}''}$$ []

[L4.3.3] Lemma (Singularity of A): For any $\underline{\lambda}'' \in W$ we have

$$A(\underline{\lambda}'')\,\underline{\lambda}'' = 0$$ []

Proof: If all costs are increased in the same proportion, then from (MCA) no allocation will change, that is

$$\underline{u}(\underline{\lambda}''+h\underline{\lambda}'') = \underline{u}(\underline{\lambda}'')$$

and since this is true for arbitrary h, the directional derivative of $\underline{u}(\underline{\lambda}'')$ in the direction $\underline{\lambda}''$ must be zero. []

[C4.3.4] Corollary: The R-dimensional function $\underline{u}(.)$ of the

R-dimensional variable $\underline{\lambda}$ is (at most) an R-1 dimensional surface (in the $\underline{u}$ space). This can also be seen from the fact that, since the scale of $\underline{\lambda}$ is arbitrary, the surface $\underline{u}$ can be described completely by fixing one component of $\underline{\lambda}$ and letting the other R-1 vary. []

[D4.3.5] Definition: The feasible region in $\underline{u}$ space is defined as

$$F \triangleq \{\underline{u} \mid 0 \leq \underline{u} \leq \underline{c}\} \qquad []$$

Since by [A4.3.1] the tangent hyperplane to $\underline{u}(\underline{\lambda})$ exists for all $\underline{\lambda} \in W$, we introduce the following concept:

[D4.3.6] Definition (Pseudo-Feasibility): $\underline{u}(\backslash)$ is Pseudo-Feasible (PF) w.r.t. F at $\underline{\lambda}=\underline{\lambda}"$, if the tangent hyperplane to $\underline{u}(.)$ at $\underline{\lambda}"$ passes through the region F (see Fig.4-1). []

In terms of this concept we have our next

[A4.3.7] Assumption: $\underline{u}(\underline{\lambda})$ is PF w.r.t. F, for all $\underline{\lambda} \in W$. []

The motivation for this is that a first-order approximation to $\underline{u}(\underline{\lambda})$ should have a non-empty intersection with the feasible region. This assumption will be considerably relaxed in sec.4.6.

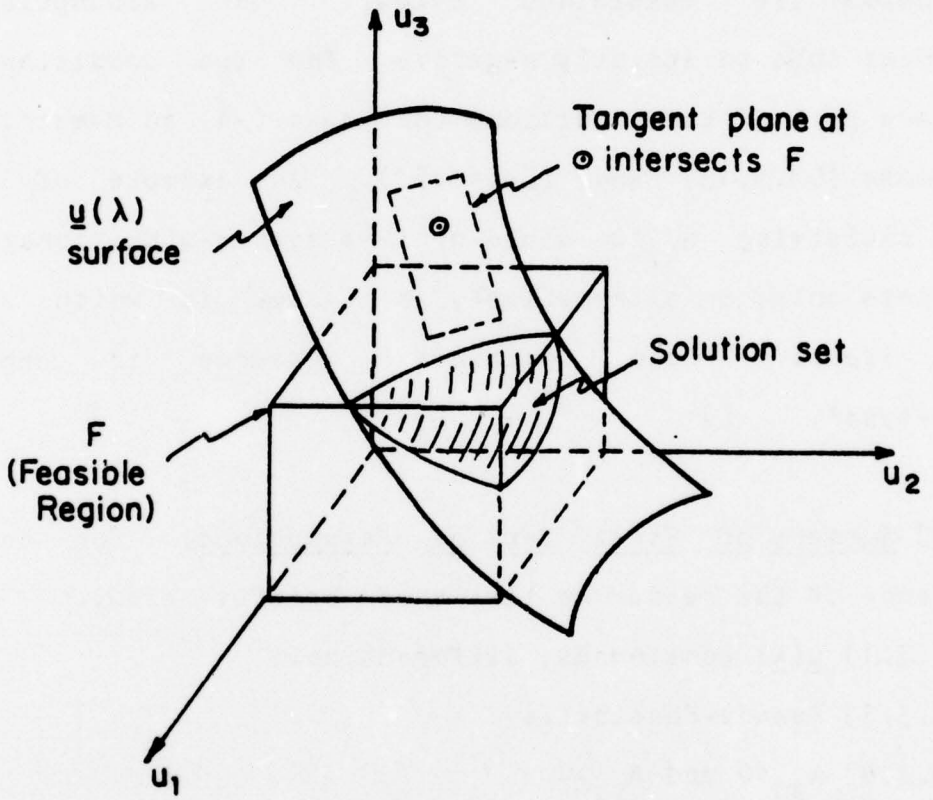


Fig.4-1: Illustration of Pseudo-Feasibility.

We shall also simplify the system configuration with [A4.3.8] Assumption: The matrix A satisfies $A_{jj}<0$ and $A_{ij}\geq 0$ $(i\neq j)$. []

[R4.3.9] Remark: (This assumption will be removed in [4.5.28] -- it is required for the Selection Algorithm only, not for our general convergence proof.) In any case A_{jj} must be non-positive (explained below). Our assumption strengthens this to strictly negative. The two conditions on A are part of the conditions that make (-A) an M-matrix (see Lemmas [L4.4.13] and [L4.4.15]). An example of a matrix satisfying $A_{ij}\geq 0$ would be in a system with storage constraints only; or alternatively a system in which we could identify one critical resource in each storage-type*. []

[4.3.10] Summary of First Set of Assumptions: For the convenience of the reader we list these briefly below.

[A4.3.1] $\underline{u}(\underline{\lambda})$ continuous, differentiable.

[A4.3.7] Pseudo-Feasibility.

[A4.3.8] $A_{jj}<0$ and $A_{ij}\geq 0$.

---

*The reason for this is as follows: From (MCA), an increase in λ_j (with all other λ_i constant) cannot cause parts not using resource j to decide to use it, and in fact it may cause some parts using resource j to move to another area. Thus we have $A_{jj}\leq 0$ and $A_{ij}\geq 0$ $(i\neq j)$.

4.4 THE SELECTION ALGORITHM

4.4.1 Outline

The purpose of this algorithm is to derive a value for $\underline{\Delta\lambda}$, the change in $\underline{\lambda}$ for the next iteration (see [4.2.2]). The algorithm does this by using the information resulting from the current iteration, i.e. $\underline{u}(\underline{\lambda})$ and $A(\underline{\lambda})$. (We assume, for the moment, that $A(\underline{\lambda})$ is known.)

Let us say we have a $\underline{\lambda}=\underline{\lambda}''$ such that $\underline{u}(\underline{\lambda}'') \notin F$. We are looking for a change $\underline{\Delta u}''$ such that

$$\underline{u}(\underline{\lambda}'') + \underline{\Delta u}'' \leq \underline{c}$$

or to a first order approximation (using [A4.3.1]), we want $\underline{\Delta\lambda}$ such that

$$\underline{u}(\underline{\lambda}'') + A(\underline{\lambda}'')\underline{\Delta\lambda} \leq \underline{c}$$

(For notational simplicity we shall drop the dependence of A on $\underline{\lambda}''$.) Defining $\underline{\Delta u}=\underline{c}-\underline{u}(\underline{\lambda}'')$ we want

[4.4.1] $$A\,\underline{\Delta\lambda} \leq \underline{\Delta u}$$

We need a method to solve this inequality, and in fact to choose from all the possible solutions for $\underline{\Delta\lambda}$. (The existence of a solution is guaranteed by [A4.3.7].) Clearly we should not, in general, try to solve the equality $A\,\underline{\Delta\lambda} = \underline{\Delta u}$. (This would attempt to move to the point where all the constraints are active, which may not even be physically realizable, and/or may not lie on the tangent hyperplane: either of these factors will cause numerical problems.) Our method is to select only certain components

of $\underline{\lambda}$" to be changed, and to look at the equation

$$A^s \underline{\Delta\lambda}^s = \underline{\Delta u}^s$$

where $\underline{\Delta\lambda}^s$ is a vector containing only the selected components of $\underline{\Delta\lambda}$, similarly with $\underline{\Delta u}^s$, and A^s is a submatrix of A containing only the rows and columns corresponding to the selected components. We solve this for $\underline{\Delta\lambda}^s=(A^s)^{-1}\underline{\Delta u}^s$. Then, in the full vector $\underline{\Delta\lambda}$, we set all selected components to their corresponding values in $\underline{\Delta\lambda}^s$, and all unselected ones are kept zero. We check to see whether $A\underline{\Delta\lambda}$ is indeed $\leq \underline{\Delta u}$. If not, we modify the selection to include other components, and repeat the procedure.

The Selection Algorithm is described in detail in the next section. Its idea is to find a solution to [4.4.1] which is satisfactory (in some sense), while at each stage exerting the minimum effort that appears necessary. For example, in a typical operation of the Algorithm with R=24, we may find we have to invert a 4x4, then 6x6 and then 10x10 matrix, at which point a suitable $\underline{\Delta\lambda}$ is found. This takes $(4^3+6^3+10^3)k = 1280k$ operations (say), whereas inverting a 24x24 matrix takes $24^3k = 13,824k$ operations.*

---

*Strictly speaking, inversion of an NxN matrix takes of the order of $N^{2.81}$ operations [A2]. However, this assumes implementation of Strassen's multiplication algorithm, which is seldom the case in practice. Therefore we assume the more usual N^3 value.

The power of the Selection Algorithm however, stems not so much from its computational savings as from its properties which are described by the next result.

[T4.4.2] Theorem 4-I

Let the Assumptions [4.3.10] hold for $\underline{\lambda}=\underline{\lambda}"$, and suppose that $\underline{u}(\underline{\lambda}") \notin F$. Then if $\underline{\lambda}">0$ we have

[T4.4.3] The Selection Algorithm (see next section) terminates before all R components of $\underline{\lambda}"$ are selected,

[T4.4.4] The $\underline{\Delta\lambda}"$ so found satisfies

$$\underline{\Delta\lambda}" = \arg \min_{\underline{\Delta\lambda} \geq 0} \{ \|\underline{\Delta\lambda}\| \mid \underline{u}(\underline{\lambda}")+A(\underline{\lambda}")\underline{\Delta\lambda} \leq \underline{c} \}$$

where $\|\underline{y}\|$ can be either the Euclidean Norm $(\sum_{i=1}^{R} y_i^2)^{0.5}$, or the l_∞ Norm $(\max_i |y_i|)$.

[]

Proof: See next section.

Remarks: There are two properties of interest in [T4.4.4] above. The first, that the Algorithm gives $\underline{\Delta\lambda}" \geq 0$. This implies that if we had $\underline{\lambda}">0$, we can be sure that $\underline{\lambda}"+\underline{\Delta\lambda}">0$. This is an important condition for the next application of the Selection Algorithm, and for several of our other results (sec.4.4.6 and 4.5). The second appealing property is that among all suitable $\underline{\Delta\lambda} \geq 0$, it finds the one of minimum norm. This will be important for our convergence proof in Theorem 4-II.

4.4.2 Description Of Selection Algorithm*

The description of our method is greatly facilitated by the concept below.

[D4.4.5] Definition: A selection vector $\underline{s}$ is defined as a vector of 0's and 1's only. Let $\underline{s}$ be an R-dimensional selection vector, with n components equalling 1, and let $J_s=\{j_1,\ldots,j_n\}$ be the set of indices of these components. For any R-dimensional vector $\underline{y}$ we define the operation of selection (#)

[4.4.6] $$\underline{s}\ (\#)\ \underline{y} \triangleq [y_{j_1},\ldots,y_{j_n}]' \qquad []$$

Conceptually, $\underline{s}(\#)\underline{y}$ "selects" only those components y_i for which $s_i=1$, and "deletes" all others. Similarly, for an RxR matrix A, $\underline{s}(\#)A$ "selects" only those elements A_{ij} for which $s_i=1$ and $s_j=1$. Thus $\underline{s}(\#)A$ is the submatrix of A obtained by deleting all rows i with $s_i=0$, and all columns j with $s_j=0$. Formally,

[D4.4.7] $$\underline{s}(\#)A \triangleq \begin{bmatrix} A_{j_1j_1} & \cdots & A_{j_1j_n} \\ \vdots & \cdots & \vdots \\ A_{j_nj_1} & \cdots & A_{j_nj_n} \end{bmatrix} \qquad []$$

---

*The details of this algorithm are not essential to the understanding of the rest of this Chapter. The reader may omit this section if desired. The properties of the algorithm in Theorem 4-I however, are important, and should be noted.

We also define a "reverse" operation to selection (for vectors only), as follows:

[D4.4.8] Definition of Distribution Operation (*): For $\underline{s}$ and n as above, and for an n-dimensional vector $\underline{z}$, we define $\underline{y}''=\underline{s}(*)\underline{z}$ to be an R-dimensional vector such that

(i) $\underline{z}=\underline{s}(\#)\underline{y}''$

(ii) If $s_j=0$ then $y''_j=0$ []

[4.4.9] Examples: $\underline{s}=\begin{bmatrix}1\\0\\1\end{bmatrix}$ $\underline{y}=\begin{bmatrix}52\\12\\18\end{bmatrix}$ $A=\begin{bmatrix}2&5&6\\4&8&3\\7&8&1\end{bmatrix}$

Then $\underline{s}(\#)\underline{y}=\begin{bmatrix}52\\18\end{bmatrix}$ and $\underline{s}(\#)A=\begin{bmatrix}2&6\\7&1\end{bmatrix}$

and $\underline{s}(*)[\underline{s}(\#)\underline{y}]=\begin{bmatrix}52\\0\\18\end{bmatrix}$.

4.4.3 Details Of Selection Algorithm

We are trying to find a $\underline{\Delta\lambda}$ satisfying the inequality [4.4.1]. Armed with our new notation, we can express concisely our algorithm for finding such a $\underline{\Delta\lambda}$ (a detailed explanation follows the algorithm).

[4.4.10] Algorithm (Selection Procedure)

DEFINE: $A = A(\underline{\lambda}")$ an RxR matrix (constant)
$\underline{\Delta u} = \underline{c}-\underline{u}(\underline{\lambda}")$ an R-vector (constant)
$\underline{s}$ = an R-vector (variable)
$\underline{\Delta\lambda}$ = an R-vector (variable)
$\underline{\Delta u}"$= an R-vector (variable)

INIT-S: for r=1 to R
do if $\Delta u_r<0$ then $s_r=1$
else $s_r=0$
end

CALC : set $\underline{\Delta\lambda} = \underline{s}(*)[(\underline{s}(\#)A)^{-1}(\underline{s}(\#)\underline{\Delta u})]$

CHECK : set $\underline{\Delta u}"=A\underline{\Delta\lambda}$
if $\underline{\Delta u}" \leq \underline{\Delta u}$ then STOP

NEXT-S: for r=1 to R
do if $s_r=1$ then nothing
else if $\Delta u"_r>\Delta u_r$ then $s_r=1$
end
go to CALC

[]

4.4.4 Explanation

In the initial stage (INIT-S) the algorithm sets $\underline{s}$ to select only those components of $\underline{\Delta u}$ which are negative, say $\underline{\Delta u}^s \hat{=} \underline{s}(\#)\underline{\Delta u}$. These components correspond to the violated constraints. It then uses the corresponding submatrix of A, say $A^s \hat{=} \underline{s}(\#)A$, to solve the linear equation

[4.4.11] $$A^s\underline{\Delta\lambda}^s = \underline{\Delta u}^s$$

The resulting $\underline{\Delta\lambda}^s$ is filled out with zeroes in the unselected components to get $\underline{\Delta\lambda}=\underline{s}(*)\underline{\Delta\lambda}^s$ in the step CALC. Now the algorithm sees whether this $\underline{\Delta\lambda}$ is satisfactory (CHECK). The method in CALC assures us that any selected component r will always satisfy $\Delta u"_r=\Delta u_r$ (proved below), so the check need only be done for the remaining components.

Any violations cause their respective components to be included in the next selection (in addition to the previously selected components) and the process is repeated. The termination of the algorithm will be proved below. First we illustrate the algorithm with a simple example.

4.4.5 Example

Let all terms be as in DEFINE of [4.4.10].

Given $\underline{\lambda}'' = \begin{bmatrix}1\\1\\1\\1\end{bmatrix}$ and $A = \begin{bmatrix}-4 & 2 & 2 & 0\\ 3 & -6 & 2 & 1\\ 0 & 1 & -2 & 1\\ 0 & 1 & 2 & -3\end{bmatrix}$, $\underline{c} = \begin{bmatrix}10\\10\\10\\10\end{bmatrix}$, $\underline{u}(\underline{\lambda}'') = \begin{bmatrix}20\\7\\5\\4\end{bmatrix}$.

Note that our example satisfies $A.\underline{\lambda}''=0$, as must be the case by [L4.3.3]. In DEFINE, we also calculate

$$\underline{\Delta u} = [-10,\ 3,\ 5,\ 6]' .$$

Then the algorithm proceeds as follows:

INIT-S: This sets $\underline{s}=[1,\ 0,\ 0,\ 0]'$.

CALC : (Terminology as in sec.4.4.4 above.) This step sets $\underline{\Delta u}^S=[-10]$, $A^S=[-4]$. Hence $\underline{\Delta\lambda}^S=(A^S)^{-1}\underline{\Delta u}^S=[2.5]$, and $\underline{\Delta\lambda}=[2.5,\ 0,\ 0,\ 0]'$.

CHECK : $\underline{\Delta u}''=A\underline{\Delta\lambda}=[-10,\ 7.5,\ 0,\ 0]$, which is not $\leq \underline{\Delta u}$ (above), so the algorithm continues.

NEXT-S: Sets $\underline{s}=[1,\ 1,\ 0,\ 0]'$.

CALC : $\underline{\Delta u}^S = \begin{bmatrix}-10\\3\end{bmatrix}$, $A^S = \begin{bmatrix}-4 & 2\\ 3 & -6\end{bmatrix}$ $\Rightarrow$ $\underline{\Delta\lambda}^S = \begin{bmatrix}3\\1\end{bmatrix}$

Hence $\underline{\Delta\lambda} = [3,\ 1,\ 0,\ 0]'$.

CHECK : $\underline{\Delta u}''=[-10,\ 3,\ 1,\ 1]'$, which is $\leq \underline{\Delta u}$, so the algorithm terminates, with $\underline{\Delta\lambda} = [3,\ 1,\ 0,\ 0]'$.

(Note that in each step CHECK above, the selected components of $\underline{\Delta u}$ exactly equal those of $\underline{\Delta u}''$, as will be proved below.)

[]

4.4.6 Proof Of Theorem 4-I

The proof is by way of a series of Lemmas.

[L4.4.12] Lemma (Equality of Selected Components): In the step CHECK of the algorithm [4.4.10] we have $\Delta u''_r = \Delta u_r$ for all r such that $s_r = 1$. []

Proof: (Notation as in sections 4.4.3 and 4.4.4 above.) Let J_s and n be as in [4.4.5] that is, $r \in J_s <=> s_r = 1$. From [4.4.11] we have, for $i \in J_s$

$$\Delta u_i^s = \sum_{k=1}^{n} A_{ik}^s \Delta\lambda_k^s = \sum_{j \in J_s} A_{ij} \Delta\lambda_j$$

Since $\underline{\Delta u}'' = A\underline{\Delta\lambda} = A[\underline{s}(*)\underline{\Delta\lambda}^s]$, we have for $i \in J_s$

$$\Delta u_i'' = \sum_{j=1}^{j=R} A_{ij} \Delta\lambda_j$$

$$= \sum_{j \in J_s} A_{ij} \Delta\lambda_j + \sum_{j \notin J_s} A_{ij} . 0 \quad \text{using [D4.4.8]}$$

$$= \Delta u_i^s \qquad []$$

For the next Lemma we shall need the following result:

[4.4.13] Hawkins-Simon Conditions* (see for example Nikaido [N2]): Consider the system of equations

[4.4.14] $$D\,\underline{\lambda} = \underline{b}$$

where D is a square matrix with nonpositive off-diagonal terms. Then the four conditions below are all equivalent**:

(I) [4.4.14] has a solution $\underline{\lambda} \geq 0$ for some $\underline{b} > 0$.

(II) [4.4.14] has a solution $\underline{\lambda} \geq 0$ for any $\underline{b} \geq 0$.

(III) The leading principal minors of D are positive.

(IV) All principal minors of D are positive.

[]

[L4.4.15] Lemma (Modified Hawkins-Simon Condition): Let D be as in [4.4.13], and define the condition

(I') The diagonal terms of D are strictly positive, and [4.4.14] has a solution $\underline{\lambda} > 0$ for some $\underline{b} \geq 0$.

Then (I') => { (I),(II),(III),(IV) } []

Proof: It is sufficient to show (I')=>(III), since the rest follows from [4.4.13]. Our proof follows Nikaido's proof of (I)=>(III) [N2] with appropriate modifications for condition (I'). Let $\underline{\lambda}$ be n-dimensional. The proof is by induction on

---

*Matrices satisfying these conditions are also known as M-matrices.

**The reader should note carefully the differences (i.e. > or $\geq$) in the various inequalities in (I),(II),(I').

n. If n=1, then (III) is true since $D_{11}>0$. Now assume that for n>1, (I')=>(III) is true for n-1. Let us subtract D_{i1}/D_{11} times the first row from the i^{th} one in [4.4.14], giving

$$[4.4.16\text{-}1] \quad D_{11}\lambda_1+D_{12}\lambda_2+\ldots+D_{1n}\lambda_n = b_1$$

$$[4.4.16\text{-}2] \quad D^*_{22}\lambda_2+\ldots+D^*_{2n}\lambda_n = b^*_2$$

$$\vdots \qquad\qquad \vdots \quad \vdots \quad \vdots \quad \vdots$$

$$[4.4.16\text{-}n] \quad D^*_{n2}\lambda_2+\ldots+D^*_{nn}\lambda_n = b^*_n$$

where (for i,j=2,...,n) we define

$$D^*_{ij}=D_{ij}-D_{i1}D_{1j}/D_{11}, \text{ so that } D^*_{ij}\leq 0, \; i\neq j$$

$$b^*_i = b_i - D_{i1}b_i/D_{11}, \text{ so that } b^*_i \geq 0.$$

This means that the n-1 dimensional system of equations [4.4.16-2] to [4.4.16-n] satisfies the conditions (I'), so that

$$\begin{vmatrix} D^*_{22} & \ldots & D^*_{2k} \\ \vdots & \ldots & \vdots \\ D^*_{k2} & \ldots & D^*_{kk} \end{vmatrix} > 0 \qquad \text{for any k, } (2\leq k\leq n)$$

Hence, by the familiar rules of determinants,

$$\begin{vmatrix} D_{11} & \ldots & D_{1k} \\ D_{21} & \ldots & D_{2k} \\ \vdots & \ldots & \vdots \\ D_{k1} & \ldots & D_{kk} \end{vmatrix} = \begin{vmatrix} D_{11} & D_{12} & \ldots & D_{1k} \\ 0 & D^*_{22} & \ldots & D^*_{2k} \\ \vdots & \vdots & \ldots & \vdots \\ 0 & D^*_{k2} & \ldots & D^*_{kk} \end{vmatrix} = D_{11} \begin{vmatrix} D^*_{22} & \ldots & D^*_{2k} \\ \vdots & \ldots & \vdots \\ D^*_{k2} & \ldots & D^*_{kk} \end{vmatrix}$$

$$> 0 \text{ for any k, } (2\leq k\leq n)$$

so that (I')=>(III) is true for n. []

[L4.4.17] Lemma: Let $A=A(\underline{\lambda}")$ where $\underline{\lambda}">0$. Let A^s be any selected matrix of A (i.e. $A^s=\underline{s}(\#)A$) of dimension $n<R$. Then for any $\underline{\lambda}^s$ of dimension n

$$A^s\underline{\lambda}^s\leq 0 \Rightarrow \underline{\lambda}^s\geq 0$$

$$\text{and} \quad A^s\underline{\lambda}^s<0 \Rightarrow \underline{\lambda}^s>0 \qquad []$$

Proof: Let us reorder the components of A such that $\underline{s}$ selects the first n. We have from [L4.3.3] that $A\underline{\lambda}"=0$. Thus

$$[4.4.18] \qquad \sum_{j=1}^{j=n} A_{ij}\lambda"_j = \sum_{j=n+1}^{j=R} (-A_{ij})\lambda"_j$$

Now from [A4.3.8] we have $A_{jj}<0$ and $A_{ij}\geq 0$ $(i\neq j)$. Thus since $\underline{\lambda}">0$, for $1\leq i\leq n$ the RHS of [4.4.18] is nonpositive, so that $A^s\underline{\lambda}"^s\leq 0$ where $\underline{\lambda}"^s\hat{=}\underline{s}(\#)\underline{\lambda}"$, or

$$[4.4.19] \qquad (-A^s)\ \underline{\lambda}"^s \geq 0$$

Thus [4.4.19] satisfies condition (I') of [4.4.15], which implies from condition (II) of [4.4.13] that for any $\underline{\lambda}^s$,

$$(-A^s)\underline{\lambda}^s \geq 0 \Rightarrow \underline{\lambda}^s\geq 0$$

which proves the first statement of the Lemma.

Now let $A^s\underline{\lambda}^s=\underline{b}<0$ (strict). This implies, a fortiori, that $\underline{\lambda}^s\geq 0$. Then for $i\leq n$,

$$A_{ii}\lambda_i^s = b_i - \sum_{j=1(j\neq i)}^{j=k} A_{ij}\lambda_j^s$$

Since $b_i<0$, $A_{ii}<0$, $A_{ij}\geq 0$ $(i\neq j)$, and $\lambda_j^s\geq 0$, we have $\lambda_i^s>0$.

[]

[L4.4.20] Lemma (Non-negativity of $\Delta\lambda$): At all stages of the Algorithm [4.4.10], provided $\underline{s}$ selects less than R components, the step CALC results in $\underline{\Delta\lambda} \geq 0$. []

Proof: Let $\underline{s}^1, \underline{s}^2, \ldots$ be the successive selection vectors tried in the algorithm. We can renumber the components of $\underline{\Delta u}$ so that

$\underline{s}^k$ selects components 1 to m^k only.

Clearly $m^1 < m^2 < \ldots$ (from [L4.4.12] and step NEXT-S). Now denote the intermediate stages of CALC as follows:

$$\underline{\Delta u}^k \hat{=} \underline{s}^k(\#)\underline{\Delta u}$$
$$A^k \hat{=} \underline{s}^k(\#)A$$
$$\underline{\Delta\lambda}^k \hat{=} (A^k)^{-1}\underline{\Delta u}^k$$

so that the step CALC can be written

$$\underline{\Delta\lambda} = \underline{s}^k(*)\underline{\Delta\lambda}^k$$

At the first selection (INIT-S) we have

$$A^1\underline{\Delta\lambda}^1 = \underline{\Delta u}^1 < 0$$

which implies $\underline{\Delta\lambda}^1 > 0$ using [L4.4.17]. Hence $\underline{\Delta\lambda} \geq 0$ for $k=1$. We will now prove $\underline{\Delta\lambda}^k > 0$ by induction on k.

Suppose $\underline{\Delta\lambda}^k > 0$, so that $\underline{\Delta\lambda} \geq 0$, but the check fails (step CHECK), leading to a new selection vector $\underline{s}^{k+1}$ (step NEXT-S), and then in CALC a new value of

[4.4.21] $$\underline{\Delta\lambda}^{k+1} = (A^{k+1})^{-1}\underline{\Delta u}^{k+1}$$

(Note that we no longer have $\underline{\Delta u}^{k+1} \leq 0$ so that we cannot use [L4.4.17] as we did above). Let

$$\underline{v} \hat{=} \underline{s}^{k+1}(\#)\,[\underline{s}^k(*)\underline{\Delta\lambda}^k]$$
$$\underline{w} \hat{=} \underline{\Delta\lambda}^{k+1} - \underline{v}$$

Then by definition of $\underline{v}$ and $\underline{\Delta u}^{k+1}$

$$j \leq m^k \Rightarrow v_j = \Delta\lambda_j^k$$

$$j > m^k \Rightarrow v_j = 0$$

$$j \leq m^k \Rightarrow \Delta u_j^{k+1} = \Delta u_j^k$$

From [4.4.21] we have

[4.4.22] $$A^{k+1}\underline{w} = \underline{\Delta u}^{k+1} - A^{k+1}\underline{v} \triangleq \underline{b} \text{ (say)}$$

For $i \leq m^k$

$$b_i = \Delta u_i^{k+1} - \sum_{j=1}^{j=m^{k+1}} A_{ij}^{k+1} v_j$$

$$= \Delta u_i^k - \sum_{j=1}^{j=m^k} A_{ij}^k \Delta\lambda_j^k$$

$$= 0 \quad \text{(by calculation of } \underline{\Delta\lambda}^k\text{)}.$$

And for $m^k < i \leq m^{k+1}$

$$b_i = \Delta u_i^{k+1} - \sum_{j=1}^{j=m^{k+1}} A_{ij}^{k+1} v_j$$

$$= \Delta u_i^{k+1} - [A(\underline{s}^k(*)\underline{\Delta\lambda}^k)]_i$$

$$= \Delta u_i^{k+1} - \Delta u_i''$$

$$< 0$$

since these components failed the check in step CHECK. We thus have $\underline{b} \leq 0$ in [4.4.22]. Now if $m^{k+1} < R$ then from [4.4.17] it follows that $\underline{w} \geq 0$. Using the strict inequality on b_i $(m^k < i \leq m^{k+1})$ above, and proceeding as in [4.4.17], we can show that $w_i > 0$ for $(m^k < i \leq m^{k+1})$. Hence

for $i \leq m^k$ $\quad \Delta\lambda_i^{k+1} = v_i + w_i = \Delta\lambda_i^k + w_i > 0$, and

for $m^k < i \leq m^{k+1}$ $\quad \Delta\lambda_i^{k+1} = v_i + w_i = 0 + w_i > 0$,

so that $\underline{\Delta\lambda}^{k+1} > 0$. []

[L4.4.23] Lemma (Termination of Algorithm): The selection algorithm [4.4.10] terminates before all R components are selected. []

Proof: First we shall prove that the algorithm cannot lead to the selection of all the R components. The proof is by contradiction.

With notation as in the preceding Lemma, suppose that we have $\underline{s}^k$ and $\underline{s}^{k+1}$ such that $m^k<R$ and

[4.4.24] $$m^{k+1} = R$$

so that $\underline{s}^{k+1}$ selects all the components. Now with

$$\underline{\Delta\lambda} \hat{=} \underline{s}^k(*)\underline{\Delta\lambda}^{k^*}$$

$$\underline{\Delta u}'' \hat{=} A\underline{\Delta\lambda}$$

we must have $\Delta u''_i=\Delta u_i$ for $1\leq i\leq m^k$ (by [L4.4.12]) and also

[4.4.25] $$\Delta u''_i>\Delta u_i \text{ for } m^k<i\leq R$$

since these components were selected in step NEXT-S.

Now, from [4.3.7] we know there exists some $\underline{v}$ such that $A\underline{v} \leq \underline{\Delta u}$. Let $\underline{w}=\underline{v}+b\underline{\lambda}''$ where b is a scalar and $\underline{\lambda}''$ is as in [L4.4.17]. Then (using [L4.3.3]) $A\underline{w}=A\underline{v} \leq \underline{\Delta u}$. Since $\underline{\lambda}''>0$, we can also choose b such that $w_R=0$.

Define $\underline{s}^{R-1}$ to select the first R-1 components. Then, with the usual notation, we have

$$A^{R-1}\underline{\Delta\lambda}^{R-1} \geq \underline{\Delta u}^{R-1} \text{ (since } \Delta\lambda_R=0, \text{ and using [4.4.25])}$$

$$A^{R-1}\ \underline{w}^{R-1} \leq \underline{\Delta u}^{R-1} \text{ (since } w_R=0)$$

$$\Rightarrow A^{R-1}(\underline{w}^{R-1}-\underline{\Delta\lambda}^{R-1}) \leq 0$$

$$\Rightarrow \underline{w}^{R-1} \geq \underline{\Delta\lambda}^{R-1} \text{ (using [L4.4.17])}$$

$$\geq 0 \text{ (since } \underline{\Delta\lambda}^{R-1} \geq 0).$$

Now let $\underline{\Delta u}^* \hat{=} A\underline{w}$. Then, using the preceding steps, and the fact that $A_{Rj} \geq 0$ for $j<R$ (from [A4.3.8]), we have

$$\Delta u_R^* = \sum_{j=1}^{j=R} A_{Rj} w_j = \sum_{j=1}^{R-1} A_{Rj} w_j > \sum_{j=1}^{R-1} A_{Rj} \Delta\lambda_j$$

$$= \sum_{j=1}^{j=R} A_{Rj} \Delta\lambda_j > \Delta u_R \text{ (using [4.4.25])}$$

which contradicts $A\underline{w} \leq \underline{\Delta u}$. So [4.4.24] cannot be true.

Since m^k is strictly increasing in k, but $m^k<R$, the algorithm must terminate. []

[L4.4.26] Lemma (Minimum Norm Property): Let $\underline{\Delta\lambda}''$ be the value of $\underline{\Delta\lambda}$ when the Algorithm [4.4.10] terminates. Then

$$\underline{\Delta\lambda}'' = \arg \min_{\underline{\Delta\lambda} \geq 0} \{ \|\underline{\Delta\lambda}\| \mid A\underline{\Delta\lambda} \leq \underline{\Delta u} \}$$

[]

Proof: Suppose the algorithm terminates when k (as in preceding Lemma) equals n. We then have found a $\underline{\Delta\lambda}$ such that $A\underline{\Delta\lambda} \leq \underline{\Delta u}$, with $A^n\underline{\Delta\lambda}^n = \underline{\Delta u}^n$ (notation as before).

Now consider any $\underline{w} \geq 0$ such that $A\underline{w} \leq \underline{\Delta u}$. Then for $i \leq m^n$,

$$\sum_{j=1}^{j=m^n} A_{ij} w_j \leq \Delta u_i - \sum_{j=m^n+1}^{j=R} A_{ij} w_j$$

$$\leq \Delta u_i \quad (A_{ij} \geq 0 \text{ for } i \neq j)$$

$$\Rightarrow A^n \underline{w}^n \leq \underline{\Delta u}^n$$

$$\Rightarrow A^n(\underline{w}^n - \underline{\Delta\lambda}^n) \leq 0$$

$$\Rightarrow \underline{w}^n \geq \underline{\Delta\lambda}^n \text{ using [L4.4.17].}$$

Also, for $i>m^n$ we have $\Delta\lambda_i=0$, so that $w_i \geq \Delta\lambda_i \geq 0$ for all $1 \leq i \leq R$, from which the minimum norm property is clear (irrespective of which of the two norms in Theorem 4-I is used). []

The last two Lemmas conclude the proof of Theorem 4-I. []

4.5 ITERATION SCHEMES FOR THE IDEALIZED PROBLEM

4.5.1 Restricted System Configuration

Retaining for the present the assumption that $\underline{u}(\underline{\lambda})$ is a continuous function of $\underline{\lambda}$ whose Jacobian $A(\underline{\lambda})$ can be calculated, and also that each area uses its own resource, we can hypothesize the following iteration scheme to solve the Artificial Problem [2.6.1]:

[4.5.1] Algorithm (SALA-1 Iteration Scheme):

INIT : Given some $\underline{\lambda}^{init} > 0$
Set $\underline{\lambda}^0 = \underline{\lambda}^{init}$
Set $k=0$

MCA : Use $\underline{\lambda}^k$ to do a Min-Cost Allocation [4.2.3] for each item, and calculate $\underline{u}(\underline{\lambda}^k)$

TEST : If $\underline{u}(\underline{\lambda}^k) \in F$ then STOP

JACOBIAN : Calculate $A(\underline{\lambda}^k)$

SELECTION: Use Selection Algorithm to calculate $\underline{\Delta\lambda}^k$

UPDATE : Set $\underline{\lambda}^{k+1} = \underline{\lambda}^k + \underline{\Delta\lambda}^k$
Set $k=k+1$
go to MCA

The next theorem will deal with the convergence characteristics of the above scheme.

4.5.2 Definitions And Assumptions

In this section we perform the groundwork necessary for a convergence analysis.

[D4.5.2] Definitions:

$$\|u\| \triangleq \max_i \frac{|u_i|}{c_i}$$

$$\|A\| \triangleq \max_i \frac{\sum_j |A_{ij}|}{c_i}$$

$$d(u,F) \triangleq \min_{\underline{u}'' \in F} \|\underline{u}-\underline{u}''\| \qquad []$$

[4.5.3] Remarks: The norms on $\underline{u}$ and A are the usual l_∞ norms, with the factor $1/c_i$ included to make the norm

independent of the scale of measurement of the ith resource. It is easily verified that with the l_∞ norm on $\underline{\Delta\lambda}$ (as in [T4.4.4]), the above definitions satisfy

[4.5.4] $\qquad \underline{\Delta u}=A\underline{\Delta\lambda} \Rightarrow \|\underline{\Delta u}\| \leq \|A\| \cdot \|\underline{\Delta\lambda}\|$

The definition of d(.,.) is the usual distance from a point to a set. We note that if $\underline{u} \notin F$

[4.5.5] $$d(\underline{u},F) = \min_{\underline{u}" \in F} \|\underline{u}-\underline{u}"\| = \max_{\{i|u_i>c_i\}} \frac{u_i-c_i}{c_i}$$

[]

[A4.5.6] Assumption: $\underline{u}(\underline{\lambda})$ is Pseudo-Feasible w.r.t. pF for all $\underline{\lambda} \in W$, where $p<1$. (In this case we say $\underline{u}(.)$ is Strictly Pseudo-Feasible, or SPF. This is a stronger restatement of [4.3.7].) []

[A4.5.7] Assumption: For all $\underline{\lambda},\underline{\lambda}" \in W$,

$$\| A(\underline{\lambda})-A(\underline{\lambda}") \| \leq D \cdot \| \underline{\lambda}-\underline{\lambda}" \| \quad (D \text{ is some constant})$$

[]

[4.5.8] Remarks: [4.5.6] assures us that the tangent hyperplane enters the interior of F, while [4.5.7] effectively bounds the second derivatives of $\underline{u}(.)$. We may draw a parallel with convergence proofs of Newton's method for finding the root of a vector function [L4,L5]. In such proofs it is customary to see [4.5.7], while our [4.5.6] is analogous to the assumption there that the gradient of the function is non-zero at the root. []

[D4.5.9] Definitions:

a. $T(\underline{\lambda}) \triangleq \min_{\underline{\Delta\lambda} \geq 0} \{ \|\underline{\Delta\lambda}\| \mid \underline{u}(\underline{\lambda})+A(\underline{\lambda})\underline{\Delta\lambda} \leq \underline{c} \}$

b. $T"(\underline{\lambda}) \triangleq \min_{\underline{\Delta\lambda} \geq 0} \{ \|\underline{\Delta\lambda}\| \mid \underline{u}(\underline{\lambda})+A(\underline{\lambda})\underline{\Delta\lambda} \leq p\underline{c} \}$

where p is the value in [A4.5.6].

c. $SEL(\underline{\lambda}) \triangleq$ any $\underline{\Delta\lambda}$ minimizing [D4.5.9a] above. (The notation reminds us that such a value is generated by the Selection algorithm.)

[]

[4.5.10] Remark: The definitions of T(.) and T"(.) are motivated by Theorem 4-I (see [T4.4.4]). We use the l_∞ norm on $\underline{\Delta\lambda}$ so that relation [4.5.4] is valid. Note that T(.) and T"(.) are scalars while SEL(.) is a vector. []

[D4.5.11] Definition: $W" \triangleq W \cap \{\underline{\lambda} \mid \underline{u}(\underline{\lambda}) \geq \underline{c}\}$ []

[4.5.12] Remark: W" is that region of W for which $\underline{u}(\underline{\lambda})$ is not in the interior of F. []

[L4.5.13] Lemma: For $\underline{\lambda} \in W"$, $T"(\underline{\lambda})$ and $\|A(\underline{\lambda})\|$ are non-zero and bounded above. []

Proof: (see next section).

[D4.5.14] Definitions:

$$\hat{a} \triangleq \sup_{\underline{\lambda} \in W"} A(\underline{\lambda})$$

$$\hat{t} \triangleq \sup_{\underline{\lambda} \in W"} \frac{T"(\underline{\lambda})}{1-p}$$ []

[4.5.15] Remark: It is in view of the previous Lemma that

we can define the above quantities, since we are assured that both quantities exist and are non-zero. []

[4.5.16] Summary of Assumptions (Second Set): The reader should compare these with the first set of Assumptions.

[A4.3.1] $\underline{u}(\underline{\lambda})$ continuous, differentiable.

[A4.5.6] Strict Pseudo-Feasibility.

[A4.3.8] $A_{jj}<0$ and $A_{ij}\geq 0$.

[A4.5.7] Bounded second derivatives.

4.5.3 Convergence Analysis

[T4.5.17] Theorem 4-II (Convergence of SALA-1 Scheme)
Let Assumptions [4.5.16] hold for some W, and let the constants D,ℓ be as already defined. For some $\underline{\lambda}^o \in W$, $(\underline{\lambda}^o>0)$ let the following conditions hold:

(i) $\|\underline{\lambda}^1-\underline{\lambda}^o\| \leq a$, where $\underline{\lambda}^1 = \underline{\lambda}^o+SEL(\underline{\lambda}^o)$

(ii) $e<1$, where $e \hat{=} a\ell D/2$

Now let $W' \hat{=} \{\underline{\lambda} \mid \|\underline{\lambda}^1-\underline{\lambda}^o\| \leq b\}$ where $b \hat{=} a/(1-e)$.
Then if $W' \subset W$, the SALA-1 scheme generates a sequence $\{\underline{\lambda}^k\}$ which converges to some $\underline{\lambda}^* \in W'$ such that $\underline{u}(\underline{\lambda}^*) \in F$.
Furthermore, the average order of convergence of the Algorithm is at least two. []

Proof: This will be via a series of Lemmas.

Proof of Lemma [L4.5.13]: This follows easily from the definitions. Let $\underline{\lambda}"=\arg\min \|A(\underline{\lambda})\|$ over $\underline{\lambda}\in W"$. Since $u(\underline{\lambda}") \geq \underline{c} > p\underline{c}$, from the SPF assumption [A4.5.6] we must have $A(\underline{\lambda}") > 0$ (or else [D4.5.9b] would have no solution). Secondly, since W (and hence W") is bounded, from [A4.5.7] the finiteness of $\|A(\underline{\lambda})\|$ ($\underline{\lambda}\in W"$) is clear.

Similarly, the SPF property assures us that for all $\underline{\lambda}\in W"$, there exists some $\underline{\Delta\lambda}$ to solve the RHS of [D4.5.9b], so $T"(\underline{\lambda})$ exists and is finite in W". Clearly, since in W" we have $\underline{u}(\underline{\lambda}) > p\underline{c}$ and $\|A(\underline{\lambda})\|$ finite, we must have $T"(\underline{\lambda})>0$. []

[L4.5.18] Lemma: For $\underline{\lambda} \in W"$, $\frac{1}{\hat{a}} \leq \frac{T(\underline{\lambda})}{d(\underline{u}(\underline{\lambda}),F)} \leq \hat{t}$ []

Remark: This will be the key Lemma for our convergence proof, since it will establish bounds on the ratio of the norm of $\underline{\Delta\lambda}$ to the distance from the feasible region. []

Proof: This relies on the minimum norm properties of the definitions above, and on the SPF assumption.

Consider some $\underline{\lambda}\in W"$ and let $\underline{\Delta\lambda}"$ be the minimizer of the RHS of [D4.5.9b]. Then letting $\underline{\Delta u}"=A(\underline{\lambda})\underline{\Delta\lambda}"$ we have*

[4.5.19] $$\Delta u_i" \leq pc_i - u_i$$

Now consider a $\underline{\Delta\lambda}"$ as a candidate for solving the RHS of [D4.5.9a], where

---

*Whenever there is no risk of confusion, we shall denote $\underline{u}(\underline{\lambda})$ simply by $\underline{u}$.

[4.5.20] $I \triangleq \{i \mid u_i - c_i > 0\}$ and $a \triangleq \inf_{i \in I}\left(\frac{u_i - c_i}{u_i - pc_i}\right)$

Clearly, $0<a<1$.

Let $\underline{u}'' \triangleq \underline{u}(\underline{\lambda}) + A(\underline{\lambda})[a\underline{\Delta\lambda}''] = \underline{u} + a\underline{\Delta u}''$.

We will show that $\underline{u}'' \in F$.

For $\{i \mid u_i - pc_i < 0\}$, $u_i'' \leq u_i + a(pc_i - u_i)$ using [4.5.19]

$\leq u_i + pc_i - u_i$ (since $a<1$)

$= pc_i < c_i$.

For $\{i \mid pc_i \leq u_i \leq c_i\}$, $u_i'' \leq u_i + a(pc_i - u_i)$ using [4.5.19]

$< c_i$ (by definition of i)

For $i \in I$, $u_i'' \leq u_i + a(pc_i - u_i)$ using [4.5.19]

$= u_i - a(u_i - pc_i)$

$\leq u_i - (u_i - c_i)$ using [4.5.20]

$= c_i$.

Thus $\underline{u}'' \in F$ and $a\underline{\Delta\lambda}''$ is a member of the RHS of [D4.5.9a]. It follows from the minimum norm property of [D4.5.9a] that

$$T(\underline{\lambda}) \leq \left\| a\underline{\Delta\lambda}'' \right\| = aT''(\underline{\lambda})$$

$$\leq \inf_{i \in I}\left[\frac{\left(\frac{u_i - c_i}{c_i}\right)}{\left(\frac{u_i - pc_i}{c_i}\right)}\right] T''(\underline{\lambda})$$

$$\leq \frac{\sup_{i \in I}\left(\frac{u_i - c_i}{c_i}\right)}{\inf_{i \in I}\left(\frac{u_i - pc_i}{c_i}\right)} T''(\underline{\lambda}) \leq \frac{d(\underline{u},F)}{1-p} T''(\underline{\lambda})$$

so that we have

$$\frac{T(\underline{\lambda})}{d(\underline{u}(\underline{\lambda}),F)} \leq \frac{T''(\underline{\lambda})}{1-p} \leq \epsilon \quad \text{from [4.5.14]}$$

which proves the RHS inequality in the Lemma.

Now let $\underline{\Delta u} \hat{=} A(\underline{\lambda})\underline{\Delta\lambda}$, for $\underline{\Delta\lambda}$ minimizing [D4.5.9a].

Then, since $\underline{u}+\underline{\Delta u} \in F$,

$$d(\underline{u},F) \leq d(\underline{u},\underline{u}+\underline{\Delta u}) = \|\underline{\Delta u}\| \leq \|A(\underline{\lambda})\| \cdot \|\underline{\Delta\lambda}\|$$

$$= \|A(\underline{\lambda})\| T(\underline{\lambda}) \leq \hat{a} T(\underline{\lambda})$$

which proves the LHS inequality in the Lemma. []

[L4.5.21] Lemma: The sequence $\{\underline{\lambda}^k\}$ in [T4.5.17] converges to some $\underline{\lambda}^* \in W'$ with $\underline{u}(\underline{\lambda}^*) \in F$. []

Proof: The existence of each $\underline{\lambda}^k$, $k>0$, is guaranteed by the SPF property provided we can show that the sequence remains in W. Assume that $\{\underline{\lambda}^k\}$ satisfies

[4.5.22] $$\|\underline{\lambda}^j - \underline{\lambda}^{j-1}\| \leq (a/e)e^{2j-1}$$

for $j=1,2,\ldots$. We then would have

[4.5.23] $$\|\underline{\lambda}^n - \underline{\lambda}^o\| \leq \sum_{j=0}^{n-1} \|\underline{\lambda}^{j+1} - \underline{\lambda}^j\| \leq (a/e) \sum_{j=0}^{n-1} e^{2j}$$

$$< a \sum_{j=0}^{n-1} e^j \quad \text{(since } e<1\text{)}$$

$$\leq a/(1-e) \hat{=} b \ .$$

Thus we would have $\underline{\lambda}^n \in W' \subset W$ for all $n>0$.

To show that [4.5.22] is true, suppose it is true for $j=1,2,\ldots,k$. Let $\underline{u}'' \hat{=} \underline{u}(\underline{\lambda}^{k-1}) + A(\underline{\lambda}^{k-1})[\underline{\lambda}^k - \underline{\lambda}^{k-1}]$, which implies $\underline{u}'' \in F$ by definition of $SEL(\underline{\lambda}^{k-1})$. Also [4.5.23] implies $\underline{\lambda}^j \in W'$ for $j=1,\ldots,k$. Let $\underline{u}^k \hat{=} \underline{u}(\underline{\lambda}^k)$. We consider the two cases $\underline{u}^k \in F$ and $\underline{u}^k \notin F$ separately.

If $\underline{u}^k \notin F$ then $\underline{\lambda}^k \in W' \subset W \Rightarrow \underline{\lambda}^k \in W''$, so using [L4.5.18] we have

$$\begin{aligned}\|\underline{\lambda}^{k+1}-\underline{\lambda}^k\| &= T(\underline{\lambda}^k) \leq \ell d(\underline{u}^k,F) \leq \ell d(\underline{u}^k,\underline{u}'') \\ &= \ell \|\underline{u}(\underline{\lambda}^k) - \underline{u}(\underline{\lambda}^{k-1}) - A(\underline{\lambda}^{k-1})[\underline{\lambda}^k-\underline{\lambda}^{k-1}]\| \\ &\leq (\ell D/2)\|\underline{\lambda}^k-\underline{\lambda}^{k-1}\|^2 \text{ using [4.5.7]} \\ &\leq (a/e)e^{2^k} .\end{aligned}$$

If $\underline{u}^k \in F$, clearly $SEL(\underline{\lambda}^k)=0$ so that the following inequality holds trivially

$$\|\underline{\lambda}^{k+1}-\underline{\lambda}^k\| = 0 \leq (a/e)e^{2^k}.$$

Thus in either case [4.5.22] holds for $j=k+1$. Now

$$\|\underline{\lambda}^1-\underline{\lambda}^0\| \leq a = (a/e)e^{2^0}$$

and so by induction it follows that [4.5.22] is true for all $j=1,2,\ldots$.

Clearly $\|\underline{\lambda}^n-\underline{\lambda}^m\| < a \sum_{j=m}^{n-1} e^j$, so that $\{\underline{\lambda}^k\}$ is a Cauchy sequence in W' and converges to some $\underline{\lambda}^* \in W'$. It remains to be shown that $\underline{u}(\underline{\lambda}^*) \in F$. From [L4.5.18],

$$d(\underline{u}(\underline{\lambda}^k),F) \leq \hat{a}T(\underline{\lambda}^k) \leq \hat{a}\|\underline{\lambda}^{k+1}-\underline{\lambda}^k\| \to 0$$

so that $d(\underline{u}(\underline{\lambda}^*),F)=0$, i.e. $\underline{u}(\underline{\lambda}^*) \in F$ by closure of F. []

[D4.5.24] Definition (Average Order of Convergence, AOC): We use the definition in [L5, p.129], adapted for vector norms. Let $\underline{\lambda}^k \to \underline{\lambda}^*$. The AOC is the infimum of the numbers $p>1$ such that

$$\limsup_{k\to\infty} \|\underline{\lambda}^k-\underline{\lambda}^*\|^{(1/p^k)} = 1$$

where "limsup" denotes limit superior. []

[L4.5.25] Lemma: The AOC of the sequence $\{\underline{\lambda}^k\}$ in Theorem 4-II is at least two. []

Proof: Consider the sequence of terms $v_k(q) \triangleq [we^{2^k}]^{(1/q^k)}$ where $0<w<\infty$, and $e<1$. We have

$$\lim_{k\to\infty} v_k(2) = \lim_{k\to\infty} w^{(1/2^k)} e = e < 1$$

while for $q>2$,

$$\lim_{k\to\infty} v_k(q) = \lim_{k\to\infty} w^{(1/q^k)} e^{(2/q)^k} = 1$$

so that $\inf_{q>1}\{q \mid \lim_{k\to\infty} v_k(q) = 1\}$ is two.

Now for the sequence $\{\underline{\lambda}^k\}$ we have

$$\|\underline{\lambda}^k - \underline{\lambda}^*\| \le \sum_{j=k}^{\infty} \|\underline{\lambda}^{j+1} - \underline{\lambda}^j\|$$

$$\le (a/e) \sum_{j=k}^{\infty} e^{2^j}$$

$$= (a/e) e^{2^k} \sum_{j=0}^{\infty} e^{2^k(2^j-1)}$$

$$< \frac{ae^{2^k}}{e(1-e)} .$$

Since we have $e<1$ and $0< \frac{a}{e(1-e)} <\infty$, by comparing the above with the results on $v_k(q)$ we deduce that the AOC of $\{\underline{\lambda}^k\}$ is at least two. []

The proof of Theorem 4-II is now complete. []

4.5.4 Remarks On Theorem 4-II

4.5.4.1 Existence Of Optimum -

The reader may ask that, since $\Delta\underline{\lambda}$ is always ≥ 0, how can we be sure that for arbitrary choice of $\underline{\lambda}^{init}$ in the algorithm [4.5.1], there exists a solution $\underline{\lambda}^* \geq \underline{\lambda}^{init}$? In Chapter 3 we gave sufficient conditions for the existence of some optimal $\underline{\lambda}$. In this Chapter we make a stronger assumption, the SPF assumption, which implicitly assures the existence of such a $\underline{\lambda}$. However, we do know that if such a $\underline{\lambda}$ exists, then there also exists some $\underline{\lambda}^* > 0$ (strictly) which is also a solution (see Corollary to Theorem 3-II). Now the scale of $\underline{\lambda}^*$ is arbitrary**, and only the relative magnitudes of the components matter (see [4.3.3] and [4.3.4]). Thus for any $\underline{\lambda}^{init}$, there exists a $b>0$ such that $b\underline{\lambda}^* \geq \underline{\lambda}^{init}$ and $b\underline{\lambda}^*$ is a solution.

4.5.4.2 Conditions In The Theorem -

These are similar to conditions used in convergence proofs of Newton's method and its extensions [L4,L5,R1]. In fact our result is similar to Robinson's [R1], but our

---

**In Chapter 3, the scale of $\underline{\lambda}^*$ was determined by the choice of the constants P^i in the Artificial Problem [3.2.5]. Since these constants are themselves arbitrary, we are not contradicting the statements in Chapter 3.

method is quite different. His proof assumes A is nonsingular, which is not so in our case (see [L4.3.3]), and he uses several properties of convex processes [R2,R3]. Our proof on the other hand depends mainly on Lemma [L4.5.18], whose proof, in turn, relies on the minimum norm properties and the SPF assumption. Intuitively, we have replaced the condition that the range of A be the whole space, by the (weaker) condition that the range of A include some point in the interior of the set $F-\underline{u}(\underline{\lambda})$.

4.5.5 General System Configuration

In the above convergence proof, the crucial properties used were the minimum norm properties in [D4.5.9]. It is seen that the restriction on the system configuration in Assumption [A4.3.8] was to ensure these properties for the selection algorithm. From the insight given us by the use of these properties we propose a general programming problem, the Minimum Norm Problem:

[4.5.26] MNP: Find $\underline{\Delta\lambda}$ minimizing $\|\underline{\Delta\lambda}\|$

a. subject to $\underline{\lambda}+\underline{\Delta\lambda} \geq \underline{\lambda}^{min}$

b. and $\underline{u}(\underline{\lambda})+A(\underline{\lambda})\underline{\Delta\lambda} \leq \underline{c}$

Here $\underline{\lambda}^{min}$ is a given strictly positive vector. This l_∞ norm problem can be solved quite simply using Linear Programming [Z2], since min $\|\underline{\Delta\lambda}\|$ is equivalent to min y subject to $y \geq \Delta\lambda_i$ and $y \geq -\Delta\lambda_i$ for all i. Thus this problem can be solved using known mech[illegible]. In that case, we can

generalize our iteration scheme:

[4.5.27] Algorithm (SALA-2 Iteration Scheme):

INIT : Given some $\underline{\lambda}^{min}>0$, and some $\underline{\lambda}^{init} \geq \underline{\lambda}^{min}$
Set $\underline{\lambda}^{o}=\underline{\lambda}^{init}$
Set $k=0$

MCA : Use $\underline{\lambda}^{k}$ to do a Min-Cost Allocation [4.2.3] for each item, and calculate $\underline{u}(\underline{\lambda}^{k})$

TEST : If $\underline{u}(\underline{\lambda}^{k}) \in F$ then STOP

JACOBIAN : Calculate $A(\underline{\lambda}^{k})$

MNP : Calculate $\underline{\Delta\lambda}^{k}$ as in [4.5.26] above

UPDATE : Set $\underline{\lambda}^{k+1}=\underline{\lambda}^{k}+\underline{\Delta\lambda}^{k}$
Set $k=k+1$
go to MCA

We shall now consider the convergence characteristics of this scheme.

[4.5.28] Third Set of Assumptions: We can considerably relax the restrictions in our second set of assumptions, to get:

[A4.3.1] $\underline{u}(\underline{\lambda})$ continuous, differentiable.

[A4.5.6] Strict Pseudo-Feasibility.

[A4.5.7] Bounded second derivatives.

[L4.5.29] Lemma: At the kth iteration of the algorithm [4.5.27], there exists some $\underline{\Delta\lambda}^{k}$ which solves MNP. []

Proof: This will be by induction. Assume that for $j=0,1,\ldots,k-1$, $\underline{\Delta\lambda}^{j}$ exists. Then, by the properties of

$\underline{\Delta\lambda}^{k-1}$,

$$\underline{\lambda}^{k} = \underline{\lambda}^{k-1} + \underline{\Delta\lambda}^{k-1} \geq \underline{\lambda}^{min}.$$

Now let $\underline{\Delta u} \hat{=} \underline{c}-\underline{u}(\underline{\lambda}^{k})$, and $A \hat{=} A(\underline{\lambda}^{k})$. From the SPF assumption, we know there exists $\underline{w}$ such that $A\underline{w} \leq \underline{\Delta u}$. Also, from [L4.3.3], $A\underline{\lambda}^{k}=0$. Hence for any scalar b,

$$A(\underline{w}+b\underline{\lambda}^{k}) \leq \underline{\Delta u} .$$

Choose b large enough so that

$$\underline{\lambda}^{k} + (\underline{w}+b\underline{\lambda}^{k}) \geq \underline{\lambda}^{min} .$$

Then $\underline{\Delta\lambda} = \underline{w}+b\underline{\lambda}^{k}$ is feasible for MNP. Thus the set of feasible solutions to MNP is non-vacuous. By closure of this set, there must exist a solution of minimum norm. Hence $\underline{\Delta\lambda}^{k}$ exists.

Now, for j=0 we have

$$\underline{\lambda}^{o} = \underline{\lambda}^{init} \geq \underline{\lambda}^{min}$$

so that by the above argument $\underline{\Delta\lambda}^{o}$ exists. Thus by induction $\underline{\Delta\lambda}^{k}$ exists for k=0,1,2,... . []

[D4.5.30] Definition: We modify [D4.5.9] to get

a. $T(\underline{\lambda})$
b. $T''(\underline{\lambda})$
c. $SEL(\underline{\lambda})$

) Let these be defined as in [D4.5.9], except that the minimizations be carried out over all $\underline{\Delta\lambda}$ such that $\underline{\lambda}+\underline{\Delta\lambda} \geq \underline{\lambda}^{min}$

[]

[L4.5.31] Lemma: The bounds in [L4.5.18] hold for the new assumptions and definitions (i.e. [4.5.28] and [D4.5.30]). []

Proof: Refer to the proof of [L4.5.18]. $\underline{\lambda}^k$ below is the same as $\underline{\lambda}$ in that proof. We showed there that $a\underline{\Delta\lambda}''$ satisfied

$$\underline{u}(\underline{\lambda}^k) + A(\underline{\lambda}^k)[a\underline{\Delta\lambda}''] \leq \underline{c} \ .$$

Now $\underline{\lambda}^k + \underline{\Delta\lambda}'' \geq \underline{\lambda}^{min}$ by [D4.5.30]

and also $\underline{\lambda}^k \geq \underline{\lambda}^{min}$ see [L4.5.29]

so $\underline{\lambda}^k + a\underline{\Delta\lambda}'' = (1-a)\underline{\lambda}^k + a(\underline{\lambda}^k + \underline{\Delta\lambda}'')$

$$\geq (1-a)\underline{\lambda}^{min} + a(\underline{\lambda}^{min})$$

$$= \underline{\lambda}^{min}$$

so that $a\underline{\Delta\lambda}''$ satisfies both the inequalities [4.5.26a and b]. The remainder of the proof is exactly as in [L4.5.18]. []

[T4.5.32] Theorem 4-III (Convergence of SALA-2 Scheme): The statements in Theorem 4-II remain valid for assumptions [4.5.28] and definitions [D4.5.30], with the SALA-2 iteration scheme. []

Proof: Theorem 4-II was proved via the following Lemmas:

[L4.5.13] Bounds on $T''(\underline{\lambda})$ and $A(\underline{\lambda})$.

[L4.5.18] Bounds on ratio $T(\underline{\lambda})/d(\underline{u}(\underline{\lambda}),F)$.

[L4.5.21] Convergence to $\underline{\lambda}^*$, with $\underline{u}(\underline{\lambda}^*) \in F$.

[L4.5.24] Order of convergence.

The reader can verify that by adding [L4.5.29] to the above set of Lemmas, and replacing [L4.5.18] by [L4.5.31], the proof of Theorem 4-III is complete. []

4.6 EXTENSION TO REALISTIC CASE

In this section we relax several assumptions, in order to extend our results to more realistic cases. First we shall find an alternative to the continuity assumption [A4.3.1]. Next we shall consider the problem of defining and calculating a suitable replacement for the Jacobian A(.). Then we shall considerably weaken the SPF assumption, to one which is likely to hold in practice. We shall use all these ideas to propose a modified iteration scheme, and then prove convergence of this scheme.

4.6.1 The Underlying Function

We now relax the unrealistic assumption [A4.3.1], that $\underline{u}(\underline{\lambda})$ is a continuous differentiable function. The sources of discontinuities in $\underline{u}(.)$ are twofold: (i) discontinuities in the usage functions of the individual items, that is $\underline{u}^i(\underline{d}^i,.)$, and (ii) an item "jumping" from one allocation to another as $\underline{\lambda}$ passes through some critical value for that item. Observe that, due to the size of the problem, the resource usage of an individual item will be quite small in comparison to the limits. Thus we postulate:

[A4.6.1] Assumption: There exists some continuous, differentiable R-dimensional function $\underline{\bar{u}}(\underline{\lambda})$, which we call the underlying function, such that

$$\frac{|\bar{u}_r(\underline{\lambda})-u_r(\underline{\lambda})|}{c_r} < e_u << 1$$

and whose Jacobian $\tilde{A}(.)$ is such that for all $\underline{\lambda},\underline{\lambda}+\underline{\Delta\lambda} \in W$

$$\frac{\|\tilde{A}(\underline{\lambda}+\underline{\Delta\lambda})-\tilde{A}(\underline{\lambda})\| \cdot \|\underline{\lambda}\|^2}{\|\underline{\Delta\lambda}\|} \leq e_Q << 1$$

where $\|A\|$ is as in [D4.5.2]. []

[R4.6.2] Remark: The subscript Q is for "Quadratic error". The terms in the above definition have been chosen so as to make e_u and e_Q dimensionless, that is independent of the scale of measurement of $\underline{\lambda}$ and $u_r(\underline{\lambda})$. (Compare e_Q with D in [A4.5.7], and remember that division by c_r is implicit in $\|A\|$.) The above assumption is reasonable in practice -- for instance, if several tens of thousands of items use resource r, then the discontinuities mentioned above will be of the order of the usage by one item, so we can expect $e_u \doteq 0.0001$. See Fig.4-2 for an illustration of $\underline{\bar{u}}(.)$. The e_Q assumption is equivalent to saying that the characteristics of the items are not bunched together, so that too many items do not change their allocations for a small change in $\underline{\lambda}$. (Without the bound on the second derivatives of $\underline{\bar{u}}$, we could always find a $\underline{\bar{u}}$ arbitrarily close to $\underline{u}$.) []

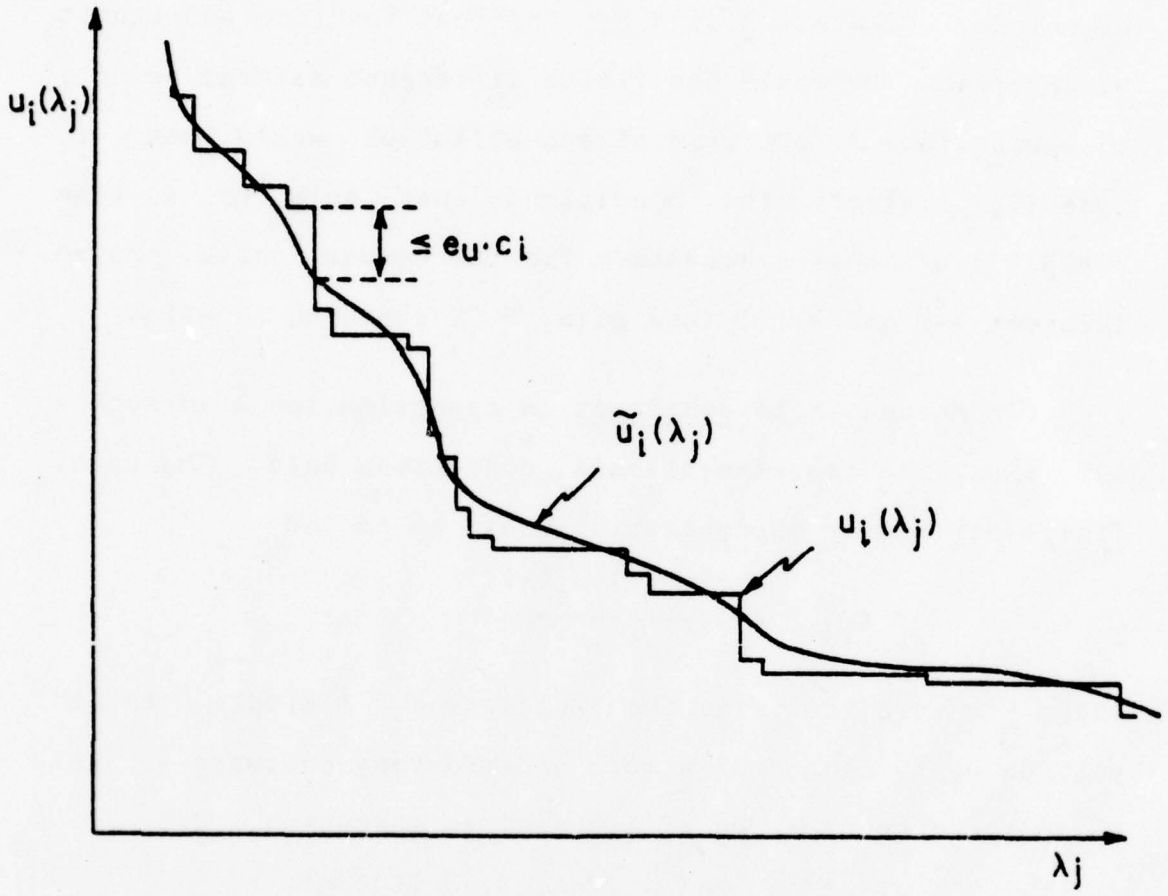


Fig.4-2: Illustration of Underlying Function.

Let λ_j vary, with $\lambda_k (k \neq j)$ constant. We can think of u_i as a function of λ_j alone, giving us a cross-section of the $\underline{u}$ versus $\underline{\lambda}$ space as above.

4.6.2 *Calculation Of Suitable Jacobian*

If we could (somehow) estimate the gradient matrix $\tilde{A}$ of the above function $\underline{\tilde{u}}(.)$, we would be able to apply our algorithm. Ofcourse $\underline{\tilde{u}}$ is a hypothetical function and cannot be observed. We could use finite difference methods on $\underline{u}(.)$ to approximate $\tilde{A}$, but such an approximation would not, in general, satisfy the conditions that held on A (see [L4.3.3]) which were necessary for the Lemmas which proved Theorems 4-I and 4-III (and also, 4-II depended on 4-I).

Our method is to <u>construct</u> an approximation $\hat{A}$ in such a way that the above-mentioned conditions hold. The usual finite-difference approximation would be to let

$$A_{ij} = \frac{u_i(\underline{\lambda}+h\underline{e}^j)-u_i(\underline{\lambda})}{h}$$

where $\underline{e}^j$= unit vector in the j direction. A simple "trick" enables us to construct a more satisfactory estimate $\hat{A}$. Let $\underline{\lambda}^k$ be the current value of $\underline{\lambda}$ during an iteration. Let

$$\underline{h}^j \hat{=} [h\lambda_1^k,\ldots,h\lambda_j^k,0,\ldots,0]'$$

$$\text{with } \underline{h}^o \hat{=} 0$$

where h is some constant, typically 0.1 . Then we calculate our estimate $\hat{A}$ by

[4.6.3] $$\hat{A}_{ij} \hat{=} \frac{u_i(\underline{\lambda}^k+\underline{h}^j) - u_i(\underline{\lambda}^k+\underline{h}^{j-1})}{h\lambda_j^k}$$

[L4.6.4] <u>Lemma</u>: The estimate $\hat{A}$ above satisfies the property of A in [L4.3.3], that is

$$\hat{A}(\underline{\lambda}^k)\,\underline{\lambda}^k = 0 \qquad []$$

$$\underline{\text{Proof}}: \quad h \sum_{j=1}^{R} \hat{A}_{ij}(\underline{\lambda}^k)\lambda_j^k = \sum_{j=1}^{R} [u_i(\underline{\lambda}^k+\underline{h}^j)-u_i(\underline{\lambda}^k+\underline{h}^{j-1})]$$

$$= u_i(\underline{\lambda}^k+\underline{h}^R)-u_i(\underline{\lambda}^k+\underline{h}^0)$$

$$= u_i(\underline{\lambda}^k+h\underline{\lambda}^k)-u_i(\underline{\lambda}^k)$$

$$= 0 \ .$$

The final step follows by the same reasoning as in [L4.3.3] -- that is, a proportional change in all the costs will not lead to a change in the allocation -- a fact which is independent of the form of the $\underline{u}(.)$ function. []

[R4.6.5] Remark: We see that the above method evaluates $\hat{A}$ at intermediate points along a path from $\underline{\lambda}^k$ to $\underline{\lambda}^k+h\underline{\lambda}^k$. This means that any given term $\hat{A}_{ij}$ is designed to estimate $\left.\frac{\partial \tilde{u}_i}{\partial \lambda_j}\right|_{\underline{\lambda}^k+\underline{h}^j}$ rather than $\left.\frac{\partial u_i}{\partial \lambda_j}\right|_{\underline{\lambda}^k}$. However, for small h, we do not expect the difference to be significant. The important point is that, although both techniques involve an estimation error, in the chosen method this error does not destroy a useful property of A. The utility of the above result is that the Selection Algorithm, if used on the estimate A, will still satisfy the statements in Theorem I (with A replaced by $\hat{A}$), and some of our existence proofs (e.g. [L4.5.29]) for Theorem 4-III also remain valid. []

In Theorem 4-IV we show that (a slightly modified version of) the SALA-2 iteration scheme will in fact converge to a solution, even in the presence of errors between the estimate $\hat{A}$ and the "underlying" gradient $\tilde{A}$. To measure these errors we have

[D4.6.6] Definition: $e_A \triangleq \max_{\underline{\lambda} \in W} \|\tilde{A}(\underline{\lambda})-\hat{A}(\underline{\lambda})\| \cdot \|\underline{\lambda}\|$

where $\|A\|$ is as in [D4.5.2]. []

[R4.6.7] Remark: The term $\|\underline{\lambda}\|$ (and the c_r implicit in [D4.5.2]) make e_A dimensionless (see remark [R4.6.2] above). []

[A4.6.8] Assumption: $e_A << 1$ []

[4.6.9] Justification: We show here that the above assumption is reasonable. First note that the scale of measurement of c_r is arbitrary, and all our definitions are independent of this scale. Thus for any given $\underline{\lambda}$, we can re-scale the c_r and λ_r (keeping $c_r\lambda_r$ constant, so that no allocation changes) to get each $\lambda_r = \|\underline{\lambda}\|$. Now $\hat{A}_{ij}$ is as in [4.6.3]. Since the second derivative of $\bar{u}$ is small, a good estimate of $\tilde{A}_{ij}$ is

$$\tilde{A}_{ij} = \frac{\bar{u}_i(\underline{\lambda}+\underline{h}^j) - \bar{u}_i(\underline{\lambda}+\underline{h}^{j-1})}{h\lambda_j}$$

Then, using [A4.6.1], $|\tilde{A}_{ij}-\hat{A}_{ij}| \leq \frac{2e_u c_i}{h\lambda_j}$

and thus $\|\underline{\lambda}\| \cdot \|\tilde{A}-\hat{A}\| \triangleq \max_i \frac{\|\underline{\lambda}\|}{c_i} \sum_{j=1}^{R} |\tilde{A}_{ij}-\hat{A}_{ij}|$

$$\leq (2e_u/h) \sum_{j=1}^{R} (\|\underline{\lambda}\| / \lambda_j)$$

$$= (2Re_u)/h \quad .$$

Thus for $e_u=0.0001$ (see [R4.6.2]), $h=0.1$, and $R=20$, we have $e_A \leq 0.02$. []

4.6.3 Relaxing The SPF Assumption

Throughout sec.4.5 we relied on the SPF assumption [A4.5.6]. If $\underline{u}^k \hat{=} \underline{u}(\underline{\lambda}^k)$ is quite far from the feasible region F, this assumption can easily be violated in practice (see Fig.4-3). This situation can be further aggravated by errors in the estimate $\hat{A}$. Let us consider an enlarged set of constraints, $\underline{c}^k \hat{=} (1+d^k)\underline{c}$ where $d^k \hat{=} d(\underline{u}^k, F)$. the factor $(1+d^k)$ is the smallest factor by which F must be enlarged in order that it contain $\underline{u}^k$ (Fig.4-4). Thus a more reasonable assumption to [A4.5.6] would be to say that the tangent hyperplane at $\underline{u}^k$ intersects $p(1+d^k)F$ for some $p<1$ (Fig.4-5). Since this enlarged region is much "nearer" to $\underline{u}^k$, we would also expect that small numerical differences between $(\underline{u}^k, \tilde{A}^k)$ and $(\underline{u}^k, \hat{A}^k)$ will not destroy the validity of this assumption (we use $\tilde{A}^k$ for $\tilde{A}(\underline{\lambda}^k)$, etc.). We now state this formally.

[A4.6.10] Assumption: (Numerical Pseudo-Feasibility, NPF) For any $\underline{\lambda}^k \in W$, the hyperplane $\underline{u}^k + \hat{A}^k \underline{\Delta\lambda}$ (for varying $\underline{\Delta\lambda}$) has a non-empty intersection with the region $p(1+d^k)F$, for some $p<1$. []

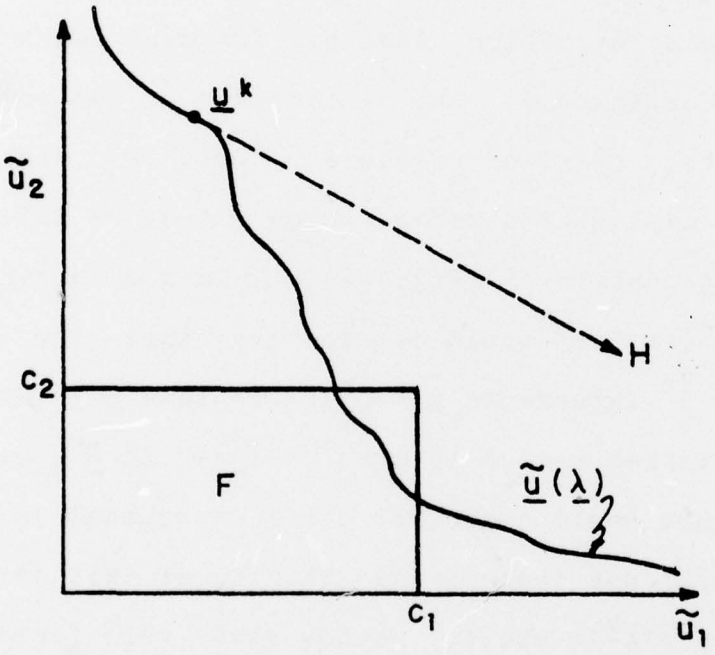


Fig.4-3: Gross violation of the SPF Assumption.
The tangent hyperplane H at $\underline{u}^k$ has no intersection with F.

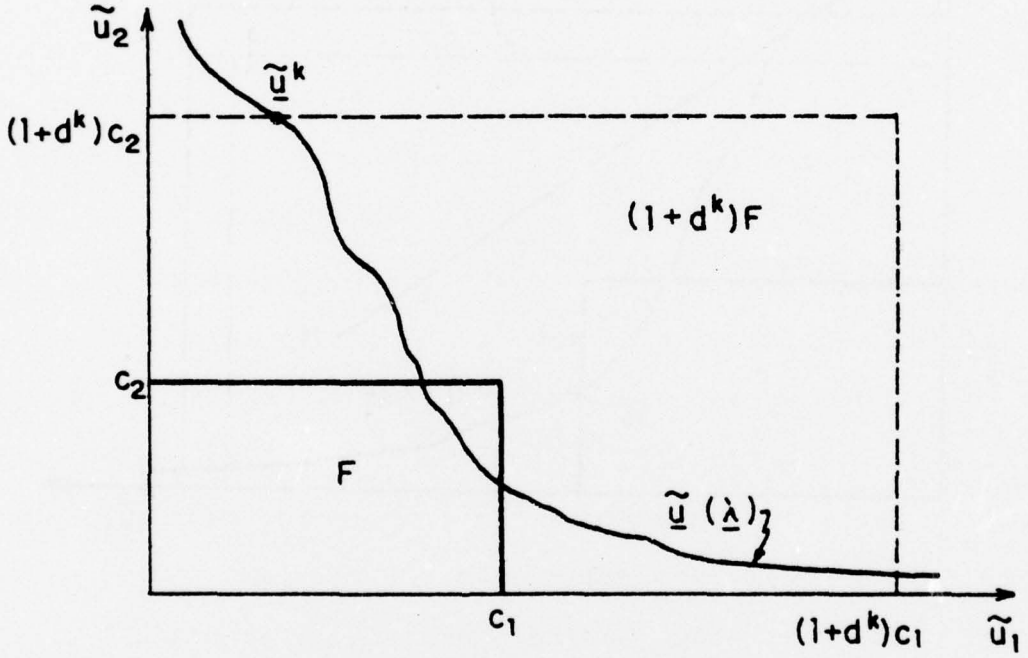


Fig.4-4: The Enlarged Region.

The region $(1+d^k)F$ just contains $\underline{u}^k$.

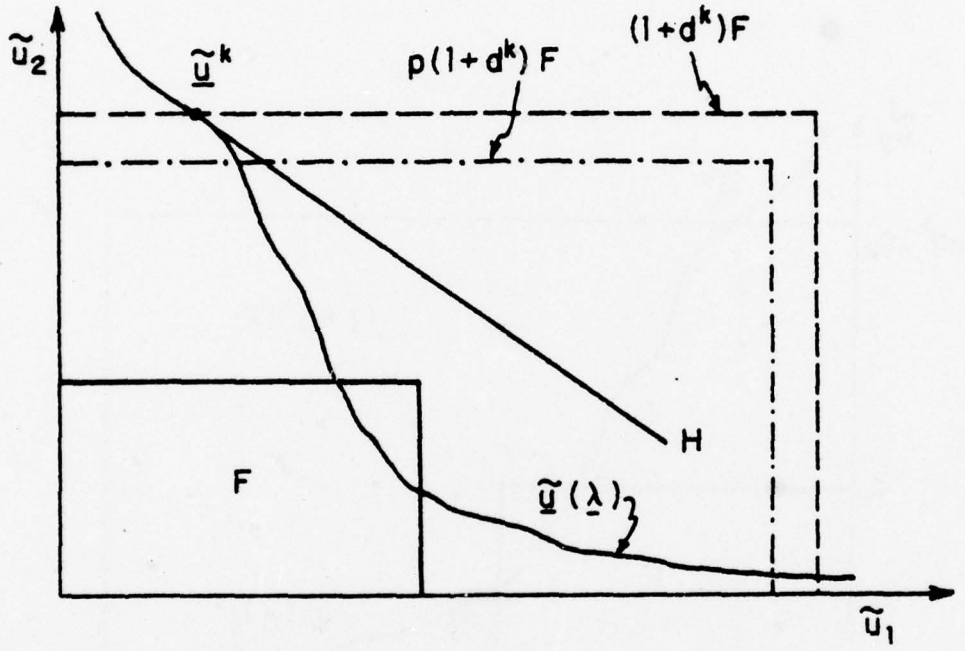


Fig.4-5: Illustration of the NPF Assumption.
It is more reasonable to expect that H will intersect the region $p(1+d^k)F$, than to expect it to intersect F.

4.6.4 Iteration Scheme For Realistic Case

We have relaxed several of our unrealistic assumptions, in order to deal with practical situations. In terms of our new concepts we have our next iteration technique.

[4.6.11] Algorithm (SALA-3 Iteration Scheme):

INIT : Given some $\underline{\lambda}^{min}>0$, and some $\underline{\lambda}^{init} \geq \underline{\lambda}^{min}$
Set $\underline{\lambda}^{o}=\underline{\lambda}^{init}$
Set $k=0$

INITMCA : Use $\underline{\lambda}^{o}$ to do a Min-Cost Allocation [4.2.3] for each item, and calculate $\underline{u}(\underline{\lambda}^{o})$
Set $\underline{u}^{o}=\underline{u}(\underline{\lambda}^{o})$

TEST : If $\underline{u}^{k} \in F$ then STOP

JACOBIAN : Calculate $\hat{A}^{k} = \hat{A}(\underline{\lambda}^{k})$ as in [4.6.3]

DISTANCE : Set $d^{k} = d(\underline{u}^{k},F)$
Set $\underline{c}^{k} = (1+d^{k})\underline{c}$

MNP : Calculate $\underline{\Delta\lambda}^{k} = \arg\min \|\underline{\Delta\lambda}\|$ subject to
$\underline{u}^{k}+\hat{A}^{k}\underline{\Delta\lambda} \leq \underline{pc}^{k}$
$\underline{\lambda}^{k}+\underline{\Delta\lambda} \geq \underline{\lambda}^{min}$

LINESRCH : Find $\hat{a} = \arg\min d(\underline{u}(\underline{\lambda}^{k}+a\underline{\Delta\lambda}^{k}),F)$

UPDATE : Set $\underline{u}^{k+1} = \underline{u}(\underline{\lambda}^{k}+\hat{a}\underline{\Delta\lambda}^{k})$
Set $\underline{\lambda}^{k+1}=\underline{\lambda}^{k}+\hat{a}\underline{\Delta\lambda}^{k}$
Set $k=k+1$
go to TEST

[R4.6.12] Remark: The numerical computation of $\hat{A}$ can be done quite efficiently. If the effort involved in evaluating the vector $\underline{u}(\underline{\lambda})$ is T units of CPU time, from [4.6.3] we see that $\hat{A}(\underline{\lambda})$ can be calculated in at most RT additional units of time. In practice, since many data items calculated for $\underline{u}(\underline{\lambda})$ can be used for calculating $\underline{u}(\underline{\lambda}+\underline{h}^{j})$, the additional effort is less than RT units of

time. We have also added a Line Search (LINESRCH) as in most implementations of Nonlinear Programming. This can also be done using well-known numerical techniques [L5]. Usually a fairly crude search suffices, and does not involve too many evaluations of $\underline{u}(.)$. The value of p has been left unspecified in the above algorithm. It can be either chosen adaptively during the algorithm, or kept as a user parameter, so that the user could "tune" the algorithm for a particular problem. (The adaptive method would start with a low value, and bring the value closer to 1 if either the MNP or LINESRCH steps were unsuccessful.)

4.6.5 Convergence Analysis.

[A4.6.13] Assumption: Let e_u, e_Q, e_A, and p be as already defined in [A4.6.1], [D4.6.6], [A4.6.10], and let

$$b \hat{=} 2e_u + e_A + (e_Q/2)$$

so that $b<<1$. Then there exists ϕ with $0<\phi<1-b-p$. []

[A4.6.14] Assumption: The $\underline{\Delta\lambda}^k$ calculated in MNP satisfies $\|\underline{\Delta\lambda}^k\| < \|\underline{\lambda}^k\|$. []

[R4.6.15] Remark: There is a relationship between the various assumptions in this analysis. In remarks following [A4.6.1] and [A4.6.8] we have already justified $b<<1$ in practice, allowing a choice of p close to 1 in [A4.6.13]

above. This makes the NPF assumption more reasonable, since it brings $p(1+d^k)F$ very close to $\underline{u}^k$. It also aids [A4.6.14] since $\underline{\Delta\lambda}^k$ will be smaller for $p(1+d^k)F$ close to $\underline{u}^k$. [A4.6.14] can be further justified. We have already remarked in [4.6.9] that it is possible to re-scale the c_r so that $\lambda_r = \|\underline{\lambda}\|$. Since in MCA, an item decides its use of resources depending on their relative costs, we can expect that a change $\Delta\lambda_j$ of the order of $\|\underline{\lambda}\|$ will cause a major change in the usage of the j^{th} resource. Loosely speaking, the step MNP only seeks to bring down the most over-used resources by a factor of p (which is close to 1). So $\|\underline{\Delta\lambda}^k\| < \|\underline{\lambda}^k\|$ is reasonable. []

[4.6.16] Summary of Assumptions (Final Set): We have considerably relaxed all our assumptions, and in addition shown why the ones below are likely to be true in practice:

[A4.6.1] Underlying function.

[A4.6.8] $e_A << 1$.

[A4.6.10] Numerical Pseudo-Feasibility.

[A4.6.13] $0 < \phi < 1-b-p$.

[A4.6.14] $\|\underline{\Delta\lambda}^k\| < \|\underline{\lambda}^k\|$.

[T4.6.17] Theorem 4-IV (Convergence of SALA-3 Scheme)
Let assumptions [4.6.16] hold. Then the SALA-3 algorithm [4.6.11] converges to a solution in a finite number of iterations. Further, if d^k is as in [4.6.11], and ϕ as in [4.6.13], then the convergence rate is characterized by

$$\text{either } d^{k+1} = 0 \text{ (i.e. solution found)}$$
$$\text{or } d^{k+1} \leq (1-\phi)d^k - \phi \qquad []$$

[R4.6.18] Remark: We note that in the worst case, the convergence rate is still better than geometric. []

Proof: First we note that, since $\hat{A}$ satisfies [L4.6.4], the existence of a $\underline{\Delta\lambda}^k$ to solve MNP at each iteration can be proved in the same way as [L4.5.29]. We develop the remainder of the proof in 4 stages:

1. The line search is replaced by a specific choice of a. Clearly, the line search in the algorithm will always do at least as well.

2. We develop an error equation relating the estimated resource usage to the actual usage.

3. We prove upper bounds on the actual usage.

4. We show that d^k decreases according to the equation above.

The details now follow.

Choice of a

From [A4.6.13] we can let $p=1-b-\phi-q$ for some $q>0$. Let us choose a such that

$$[4.6.19] \qquad \frac{2e_u+\phi}{2e_u+\phi+q} \leq a < 1 .$$

From this we deduce that

$$a(2e_u+\phi+q) \geq 2e_u+\phi$$

$$\Rightarrow \quad a(2e_u+1-p-b) \geq 2e_u+\phi$$

$$\Rightarrow \quad a(1-p-e_Q/2-e_A) \geq 2e_u+\phi \quad \text{using [A4.6.13]}$$

and thus

$$[4.6.20] \qquad a(1-p) - \phi \geq 2e_u+ae_A+ae_Q/2$$

an equation we shall use later in this proof.

Error Equation

Write $\underline{u}$ for $\underline{u}(\underline{\lambda})$ and $\underline{u}^a$ for $\underline{u}(\underline{\lambda}+a\underline{\Delta\lambda})$, for simplicity. Define $\underline{\Delta u} \hat{=} \hat{A}(\underline{\lambda})\underline{\Delta\lambda}$, so that $\underline{u}+\underline{\Delta u} \leq p\underline{c}^k$ by MNP. Also let $y \hat{=} \|\underline{\Delta\lambda}\|/\|\underline{\lambda}\| < 1$ by [A4.6.14]. Then we have

$$u_i^a \hat{=} u_i(\underline{\lambda}+a\underline{\Delta\lambda}) \leq \bar{u}_i(\underline{\lambda}+a\underline{\Delta\lambda})+c_i^k e_u \quad \text{using [A4.6.1] and } \underline{c}^k>\underline{c}$$

$$\leq \bar{u}_i(\underline{\lambda})+[\tilde{A}(\underline{\lambda})a\underline{\Delta\lambda}]_i + c_i^k \frac{e_Q\|a\underline{\Delta\lambda}\|^2}{2\|\underline{\lambda}\|^2} + c_i^k e_u \quad \text{using [A4.6.1]}$$

$$\leq \bar{u}_i(\underline{\lambda})+[\hat{A}(\underline{\lambda})a\underline{\Delta\lambda}]_i+[\tilde{A}(\underline{\lambda})-\hat{A}(\underline{\lambda})]a\underline{\Delta\lambda}+(ay)^2(e_Q/2)c_i^k+e_u c_i^k$$

$$\leq \bar{u}_i(\underline{\lambda})+a\Delta u_i+e_A c_i^k(ay)+(ay)^2 c_i^k e_Q/2+e_u c_i^k \quad \text{using [D4.6.6]}$$

$$\Rightarrow \quad \frac{u_i^a}{c_i^k} \leq \frac{u_i}{c_i^k} + a\frac{\Delta u_i}{c_i^k} + 2e_u+e_A(ay)+(e_Q/2)(ay)^2 \quad \text{using [A4.6.1]}$$

$$\leq \frac{u_i}{c_i^k} + a\frac{\Delta u_i}{c_i^k} + 2e_u + e_A + e_Q/2 \qquad \text{since } a<1, y<1$$

so that

$$[4.6.21] \qquad \frac{u_i^a}{c_i^k} \leq \frac{u_i}{c_i^k} + a\frac{\Delta u_i}{c_i^k} + a(1-p)-\phi \qquad \text{using [4.6.20]}$$

which is our desired error equation.

Bounds on the u_i

First we prove a minor result:

[4.6.22] $a<1$ and $u_i<c_i^k \Rightarrow u_i+a(pc_i^k-u_i) < c_i^k-a(1-p)c_i^k$ []

Proof:

$$a(c_i^k-u_i) < c_i^k-u_i$$

$$\Rightarrow \quad a(c_i^k-pc_i^k-u_i+pc_i^k) < c_i^k-u_i$$

$$\Rightarrow \quad a(1-p)c_i^k+a(pc_i^k-u_i) < c_i^k-u_i$$

$$\Rightarrow \quad u_i-a(pc_i^k-u_i) < c_i^k-a(1-p)c_i^k \qquad []$$

Now consider two sets of indices:

$$J \triangleq \{i \mid u_i<c_i^k\} \quad \text{and} \quad K \triangleq \{i \mid c_i^k \leq u_i\} \ .$$

We have $\Delta u_i \leq pc_i^k-u_i$ (by step MNP) so that $a\Delta u_i \leq a(pc_i^k-u_i)$.

From [4.6.21], for $i \in J$,

$$\frac{u_i^a}{c_i^k} \leq \frac{u_i+a(pc_i^k-u_i)}{c_i^k} + a(1-p) - \phi$$

$$< 1 - \phi \qquad \text{using [4.6.22]}$$

and for $i \in K$, since $u_i \geq c_i^k$,

$$\frac{u_i^a}{c_i^k} \leq \frac{u_i+a(pc_i^k-c_i^k)}{c_i^k} + a(1-p) - \phi$$

$$= \frac{u_i}{c_i^k} - \phi \ .$$

Equation for d^k

Note that from [D4.5.5] we have

$$\frac{u_i}{c_i} - 1 \leq d^k$$

and from the algorithm we have

$$\underline{c}^k = (1+d^k)\underline{c} \ .$$

Consider the sets J, K of the previous section, along with their bounds on u_i^a.

For $i \in J$ $\quad \dfrac{u_i^a}{c_i} < (1-\phi)\dfrac{c_i^k}{c_i} = (1-\phi)(1+d^k) = (1+d^k) - \phi(1+d^k)$.

For $i \in K$ $\quad \dfrac{u_i^a}{c_i} \leq \dfrac{u_i}{c_i} - \phi\dfrac{c_i^k}{c_i} \leq 1+d^k - \phi(1+d^k)$

so that for all i we have

$$\frac{u_i^a}{c_i} - 1 \leq (1-\phi)d^k - \phi$$

And note that $d(\underline{u}^a, F) = \max_{\{u_i^a > c_i\}} \left[\dfrac{u_i^a}{c_i} - 1 \right]$

Identifying $\underline{u}^k$ with $\underline{u}$ and $\underline{u}^{k+1}$ with $\underline{u}^a$, we thus have

$$d^{k+1} \leq (1-\phi)d^k - \phi$$

as long as the RHS is non-negative. Clearly, $d^{k+1}=0$ will be achieved in a finite number of iterations, at which point the algorithm will stop.

This concludes the proof of Theorem 4-IV. []

4.7 REVIEW OF RESULTS AND COMPARISON WITH OTHER WORK

Starting with a simplistic model, we proved some properties of the Selection Algorithm, a means for generating $\underline{\Delta\lambda}^k$. Using these properties we were able to prove quadratic convergence of an iteration scheme for a restricted system configuration. We then generalized our method of computing $\underline{\Delta\lambda}^k$, and proved quadratic convergence of an iteration scheme for a general system configuration. Next we incorporated the actual functions in our model, replacing "local" assumptions (continuity) by "global" assumptions (system structure and bounds on certain error terms -- it was shown that these bounds are reasonable for a large system). We were then able to prove convergence of an algorithm operating on these functions.

The use of the "underlying function" assumption [A4.6.1] seems to be a powerful way to deal with local discontinuities. Katkovnik [K1] and Ermol'ev [E3] have suggested using Stochastic Approximation techniques for "smoothing out" such local discontinuities. (Conceptually, his method tries to identify our underlying function.) His method however, would involve a large number of function evaluations* to compute each row of $\hat{A}$; our method requires only one new evaluation for each row (see [R4.6.12]).

---

*Each function evaluation involves doing an allocation of all parts.

Our assumptions, specially for Theorem 4-IV, may seem strict and somewhat arbitrary. However, we are placing almost no restrictions on the individual Usage functions $\underline{u}^i(\underline{d}^i, \underline{x}^i)$. If we used Katkovnik's approach, along with a general framework for the analysis of Recursive Stochastic Approximation Algorithms,(e.g. Ljung [L6] or Kushner [K2]) we would require strict conditions on the differentiability of various functions and bounds on their derivatives. As above, one may think of our "global" assumptions as replacing such "local" assumptions.

The final test of the validity of our assumptions is ofcourse use of the algorithm in actual cases. We have used the Algorithm successfully on numerous problems, both for evaluating design of new Warehouse facilities, and for improving the operation of existing facilities. Examples of this will follow in the next Chapter.

CHAPTER 5

THE DESIGN OF A PRACTICAL RESOURCE MANAGEMENT SYSTEM

5.1 OVERVIEW OF CHAPTER

The previous Chapter dealt with the "Initial Allocation Problem". In this Chapter we illustrate how, in an operational system, we implement the solution to this problem, and how our approach enables simple and efficient solution of both the "New Assignment Problem" and the "Periodic Review Problem" (see Chapter 1 for the definition of these problems).

We shall also suggest some additions to the algorithms of Chapter 4, which enhance their operation in practice. Most notable among these is the incorporation of a "Cost of Reallocation" for each part.

The main features of the computerized Resource Management System implemented in the FIAT Warehouse are then described, including some details of the program package and its performance. We conclude with a representative example of the use of the program to solve a typical design-evaluation problem for the warehouse.

5.2 CONCEPTS OF THE PRACTICAL SOLUTION

5.2.1 Approach Used

The first step in the practical solution is deciding on the calculation of the estimated resource usages, that is, the functions $\underline{u}^i(\underline{d}^i,\underline{x}^i)$. For each part i, these functions should be designed to estimate the current mean value of $\underline{u}^i$ for this part. An example of such a calculation is given in Appendix 5A. The estimate of total usage of the rth resource, i.e.

$$u_r = \sum_{i=1}^{I} u_r^i(\underline{d}^i,\underline{x}^i)$$

will then be the sum of these means.

Remark: Although the variation in the individual part's resource usages u_r^i about their mean may be quite high from day to day, the variation in the sum u_r will be much less in comparison to the value of u_r itself.*

The Initial Allocation Problem is solved with the value of each constraint c_r set at some proportion p_r of the actual physical capacity of that resource, for example 70% of crane-time or 90% of storage capacity. The value of p_r,

---

*Formally, the standard error (=standard deviation/mean) of the sum of N identically distributed independent random variables is $1/\sqrt{N}$ times the standard error of any one of the random variables.

for each resource r, is set by management to reflect a certain safety margin. For instance, by analysing the history of demand on a particular resource, a reasonable safety margin could be set.

The New Parts Problem is solved for each new part simply by doing a Minimum Cost Allocation for that part, using the current costs $\underline{\lambda}$. In other words, we solve equation (MCA) [4.2.3] for that part. Note that the allocation for each new part can be made by solving this (relatively) simple problem independently of the rest of the parts.

For the Periodic Review Problem, we note that:

1. The allocation of new parts will cause a gradual addition to the (average) usage of various resources.

2. The changing characteristics of each part will cause changes (increases or decreases) in the (average) usage of each resource.

3. We note that the changes above are expected to be small with respect to the constraints. It is physically possible to continue operation due to the safety margin in each limit value.

4. The net result for each resource, will be a gradual increase or decrease in the average total usage of that resource.

The aim of the periodic review is to track these averages, and to keep them below a specified value (say c_r+d_r, where d_r is the maximum deviation allowed for resource r). This can be done in as sophisticated a manner as desired. For instance, in the Volvera warehouse we found that a monthly re-allocation was sufficient. In a more rapidly changing environment, a daily, or even hourly reallocation could be done.

The reallocation is done by using the latest $\underline{\lambda}$, say $\underline{\lambda}^{old}$, as a starting point for iterations of the SALA algorithm. Since the data vectors $\underline{d}^i$ for each part have changed, and since new parts have been added, $\underline{u}(\underline{\lambda}^{old})$ which was feasible earlier, will not necessarily be feasible now. However, it is likely that a small change in $\underline{\lambda}^{old}$ (achieved in one iteration of the SALA algorithm, say) will make it feasible. The new value, say $\underline{\lambda}^{new}$, will now be used for allocation of each part, and for the allocation of new parts. This method can be further enhanced as described below.

5.2.2 Cost Of Re-allocation

We now describe the incorporation of a feature called the "cost of re-allocation" for each part. In the terminology of the previous section, for any $\underline{\lambda}^{new}$, we only change the allocation of a part if its total (new) resource usage cost plus a certain re-allocation cost, is less than the existing (old) resource usage cost, that is:

Let $\underline{x}^i_{new}$ = arg min (MCA) for given $\underline{\lambda}^{new}$

$\underline{x}^i_{old}$ = current allocation of part i

$\underline{u}^i_{new} = \underline{u}^i(\underline{d}^i, \underline{x}^i_{new})$ [$\underline{d}^i$ is the current (new) data]

$\underline{u}^i_{old} = \underline{u}^i(\underline{d}^i, \underline{x}^i_{old})$

$C(\underline{x}^i_{old}, \underline{x}^i_{new})$ = cost of re-allocation from $\underline{x}^i_{old}$ to $\underline{x}^i_{new}$

Then we reallocate the ith part from $\underline{x}^i_{old}$ to $\underline{x}^i_{new}$ only if

[5.2.1] $$\underline{\lambda}^{new}, \underline{u}^i_{new} + C(\underline{x}^i_{old}, \underline{x}^i_{new}) < \underline{\lambda}^{new}, \underline{u}^i_{old} .$$

The calculation of C(.,.) will be discussed below.

The addition of this reallocation cost has many advantages:

1. It makes the allocation of each part "stable", that is, for small changes in $\underline{\lambda}$, we do not find ourselves moving the same part back and forth (or else we might spend all the warehouse resources in constantly reallocating parts!)

2. It makes the decision $\underline{x}^i$, for each new part, valid in the long term, and hence a good decision according to the criterion in Chapter 2, Section

4.3. The reason for this is that usually very small changes in $\underline{\lambda}$ are sufficient to keep the total resource usages within their margins as above. Thus, as just described, the allocation of each new part is likely to remain unchanged for some time. Only when the characteristics ($\underline{d}^i$) of this part change significantly, or when $\underline{\lambda}$ has changed significantly, will it be likely that this part will be reallocated, to reflect the new situation.

3. The reallocation cost can be incorporated directly in the SALA Algorithm during its iterations, so that its decision $\underline{x}^i$ for each part, and its calculation of total resource usage $\underline{u}(\underline{\lambda}^{new})$, will be made using the rule [5.2.1] above. Thus the $\underline{\lambda}^{new}$ found by the algorithm will take into account the reallocation costs.

4. The cost calculation C(.,.) can be modelled to reflect the physical work necessary to change from one allocation to another. (This would require some "discounting" calculation, to trade-off between a one-time cost and a daily operating cost over some time horizon.)

5. The calculation of C(.,.) can also be made to reflect management preferences over and above physical work in the above paragraph. For example, if management is trying to gradually shut down a

certain storage-type, the cost of moving out of it can be set lower than the physical cost, whereas the cost for moving into it can be set very high.

We note here that the modification of the SALA algorithm described in this section is a heuristical one, whose consequences have not been studied theoretically. However, it appeals to common sense, and has definitely proven its worth in practice. The method used for calculation of C(.,.) in the Volvera warehouse will be described in sec.5.3.7.

5.3 FEATURES OF THE VOLVERA IMPLEMENTATION

5.3.1 Introduction

The SALA program package, whose features are described in this section, was designed and implemented by the author, in connection with the C.S.Draper Lab., for use by FIAT Ricambi at their Central warehouse in Volvera, Italy. This program package provides management with a versatile tool with which to tackle the Storage Allocation and Resource Management (SARM) task at Volvera (see Chapter 2 for a description of this task). The concepts of SALA, from a management point of view, are described in the study [F2], and the details of its implementation are contained in the SALA Reference Manual [S3]. A brief description of the SALA

program package follows below.

SALA is designed to provide Volvera with three main functions for the SARM task:

1. Periodic Review and Reallocation

2. New Parts Allocation

3. Design Evaluation

The first two of these have been described in Chapter 2, and their solution is along the lines of sec.5.2.1 above. The third is an important additional feature, described next.

5.3.2 SALA As A Design Evaluation Tool

The power of the SALA package lies in its flexibility. It has been designed to enable management evaluate any future warehouse modifications, within the framework of a set of known possible modifications. (The modularity of the package means this set can be expanded in the future, if necessary, by an experienced FORTRAN programmer.) In effect, SALA provides the user with the building blocks with which to define his design, and also with the tool that will then evaluate this design. This enables the study of future modifications to the warehouse, such as closing down an existing storage-type, or adding new storage-types. A detailed illustration of this will be given in sec.5.4.

We find that the FIAT management at Volvera are making extensive use of this feature for evaluating future designs, and consider this aspect of SALA a valuable decision-making aid.

5.3.3 Hierarchy Of Decision-Making

We describe here an important feature of the SARM project. The SALA algorithm is designed to make long-term decisions involving alocations of parts among different storage-types. Now these storage-types are themselves fairly large systems, and their day-to-day operation is controlled by Real-Time computerized algorithms. Other portions of the project have studied the optimization of operations at this level. Thus we have a natural hierarchy of decisions. The operating strategies within a storage-type give rise to certain average measures of performance in that storage-type (e.g. crane-time per pick). These measures are then used by the SALA algorithm for its decisions between storage-types. At a yet higher level, the strategy of the SALA algorithm generates certain performance measures for the Volvera warehouse as a whole, and these are used by another project, the Inventory Control project, to decide on the distribution of stock at Volvera and at other sites throughout the world.

5.3.4 Some Program Details

The SALA package is written entirely in FORTRAN.* It is highly modularized, for ease of modification, debugging, and implementation on different computers.** It consists of some 40 modules totalling some 7000 lines of FORTRAN. Less than 30% of this is the algorithm code. The remainder is for several different options, for generating various statistics on each allocation, and for detailed diagnostics if required for checking the operation of the program.

The package is currently implemented on a Honeywell 6600 computer. The typical CPU time for an iteration involving all 60,000 parts is about one hour. (An iteration, as described in Chapter 4, involves calculation of the Jacobian, as well as a step-size calculation.) Using a sampling technique, described in the next section, it is possible to solve an initial allocation problem in a few hours of CPU time. For an incoming new part, a new allocation decision takes only a few milliseconds, and can be done in real-time. A periodic review usually requires

---

*The output generated by SALA is further analyzed and tabulated for warehouse personnel, using COBOL programs.

**In fact, the package was developed and tested on a DEC PDP-11/70 at C.S.Draper Lab. (Cambridge, Mass.) before implementation on the Honeywell H6600 in Italy.

only one iteration, or one hour of CPU time.

This amount of time should be compared with the File Allocation Program of Mahmoud and Riordan (see Table 6-I, in Chapter 6) which takes 16 minutes of CPU time on an XDS Sigma 7, for a problem with 20 files and 5 nodes. Our solution is for a problem with 60,000 items and 16 storage-types. We can also compare SALA's performance with the previous techniques used at Volvera. These involved sorting all the parts into various subsets according to their characteristics (this took hours of CPU time), and then manually deciding on different subsets of parts to be reallocated, to meet a large number of criteria (this took months of man-time). In either case SALA's performance compares very favorably.

5.3.5 The Sampling Technique

This technique brings the algorithm from an arbitrary initial $\underline{\lambda}$, say $\underline{\lambda}^{o}$, to a nearly optimal $\underline{\lambda}$, in an efficient manner. A random subset of parts is selected, and all constraints are reduced proportionally. For example, if 10% (i.e. 6000) parts were selected, the constraints would be set at 10% of their actual values. The algorithm is then made to iterate on this sample until a solution, say $\underline{\lambda}^{*}$, is found. Usually, 5 to 15 iterations suffice, and since this sample is a tenth of the whole sample, 10 iterations can be done in an hour. Now, because the sample is a

representative cross-section of all the parts, the value $\underline{\lambda}^*$ is a very good starting point for iterations on the whole sample, and usually one or two iterations on the whole sample suffice. This means that the entire problem can be solved in 3 to 4 hours of CPU time, as mentioned above. The sampling technique is another way in which decentralized solution of the problem proves to be advantageous.

5.3.6 Solution Of MCA

Each iteration of the SALA algorithm involves solving the Minimum Cost Allocation problem (MCA) [4.2.3] for each part. The solution to this problem can, in general, be designed to fit the needs of the specific implementation. For the Volvera problem, our solution is described below. A rather different example of solving (MCA) is given in Chapter 6 for the File Allocation Problem.

In Chapter 4, we noted that we would restrict each $\underline{x}^i$ to a discrete strategy set. In the Volvera warehouse, a review of management and operating requirements, plus the possible complexity of solving (MCA) brought us to the decision that each part would be stored in one storage-type only. (Some exceptions to this are described below.) The rationale for this can be stated as: "If it is possible to find a feasible solution with this restriction, then we have achieved our goal anyway, and with a simpler strategy. If we do not achieve a feasible solution, then we will expand

our strategy set."

If we were to use integer programming, this restriction would still mean that we have a 1-million dimensional 0-1 integer programming problem, with the added 60,000 constraints

$$\sum_{s=1}^{16} x_s^i = 1 \quad \text{for each } i \ .$$

However, for the problem (MCA), this restriction makes the solution trivial, and it can be found by a simple "loop" over the 16 storage-types. Let

$$\underline{e}^j = \text{unit vector in } j \text{ direction}$$

and let

$$t = \arg \min_{[s=1,\ldots,16]} \underline{\lambda}' \underline{u}^i (\underline{d}^i, Q^i \underline{e}^s) \ .$$

Then for the ith part, the optimal allocation for a given $\underline{\lambda}$ is

$$\underline{x}^i(\underline{\lambda}) = Q^i \underline{e}^t \ .$$

In other words, allocate all the quantity to the storage-type where the total resource usage cost is a minimum.

For operational reasons, it is necessary for some parts to be stored in two storage-types. This ability is incorporated in SALA by pooling certain resources of the two storage-types, and letting SALA find an allocation satisfying those pooled constraints. A second program, designed to satisy other management requirements, and which operates on a more frequent basis, then decides on the

distribution of containers between the two areas. This provides a convenient hierarchical separation of decision-making, similar to that described in sec.5.3.3 above. For details, see Suri [S4].

5.3.7 Calculation Of Reallocation Cost

The calculation of C(.,.) is done empirically in each review period, by allowing warehouse personnel to examine detailed "actual/recommended" tabulations, giving the cost-improvement $\underline{\lambda}'(\underline{u}^i_{old} - \underline{u}^i_{new})$ for each part. (These tabulations compare the actual allocation with SALA's recommended re-allocation). The tables are sorted into subsets by current and recommended allocation, and each subset ordered by cost-improvement. By choosing a threshold value for cost-improvement for each subset of tabulations, warehouse personnel are able to control the number of reallocations between any two storage-types in each period. Their decisions are based on their current priorities and the workload in the warehouse. The tabulations are also useful when carrying out reallocations: parts with the greatest cost-improvement are always reallocated first, since they represent the greatest mismatch between current allocation and part characteristics.

5.4 EXAMPLE OF A DESIGN-EVALUATION PROBLEM

We illustrate the use of SALA at Volvera with a typical design-evaluation problem.* This involves selection of parts to be allocated to a proposed new area, at the same time reallocating parts between existing areas to meet future requirements.

The Volvera management is considering building a new area at Volvera, called the MPM ("Magazzino Prelievo Manuale", or manual picking warehouse), with very high picking capacity, but only a limited number of parts can be assigned there. The SALA program is being used to evaluate various design specifications for the MPM, by looking at the effect of certain decisions on the operating statistics of the whole warehouse.

A typical design specification is given in Table 5-I. This specification is for 5000 part-numbers and 12 storage-types. There are 17 constrained resources (see the first column). For example, the entry "M-11 Part Nos" refers to the number of Part-numbers in storage-type "M-11". This is a storage-type in the MPM, and it has a limit on the number of parts it can handle, as mentioned above. Some resources are shared, for example (1+M)-11 means a resource

---

*This example uses a sample of 5000 parts, so as to maintain confidentiality of the detailed operating statistics of the FIAT warehouse.

used by both storage-type 1-11 and storage-type M-11. (The meaning of these codes is not relevent to this explanation.)

The next two columns in the Table, "Usage Today" and "Usage Limit" reflect the current operating values versus the design specifications laid down by management. The number of Part-Nos in M-11, M-20, M-60 (all in the MPM) are currently zero, and their design capacities are given as 291, 353, and 20, respectively. In other resources, we see that management would like to see some decreases in the usages -- most notably in "1-20 Crane-Minutes" which exceed the desired value by 500%. (The sign ">" is used in the Table to flag over-used resources.)

The problem of finding an allocation to meet the future capacities was solved using the SALA program. It was found that for the specified capacities, a solution did indeed exist, and the resource usages of this solution are given in the fourth column of the table. The main point to note is that after reallocation, we find very little spare capacity in most resources, showing that the problem as specified is not an easy one to solve. (Resources whose future usage is up to their design capacity are marked with "=" or "≡" in the table.)

The costs $\underline{\lambda}$ computed by SALA for each resource are shown in the final column. If the MPM were ready, and parts had been assigned as recommended by SALA, then new parts arriving at the warehouse would be assigned using these

costs to compute their best allocation. Also, for the design problem, if management decided to try a different design, these costs would be a good starting point for the next set of iterations.

Table 5-I: Design Evaluation Example

RESOURCE NAME	USAGE TODAY		USAGE LIMIT		--SALA-ALLOCATION-- USAGE	COST
M-11 Part Nos	0		291	=	289	2798
M-20 Part Nos	0		353		340	4178
M-60 Part Nos	0		20	=	20	17498
Q-11 Part Nos	349	>	292	=	292	2756
Q-20 Part Nos	63	>	50	=	50	4958
Q-60 Part Nos	7		7	=	6	17383
(1+M)-11 Shelf Slots	2341		2837		2334	255
(1+M)-20 Shelf Slots	3400		4201	=	4200	1109
(1+M)-60 Shelf Slots	2091		2177	=	2171	3105
1-(11+60) Crane-mins	516	>>	190		167	11385
1-20 Crane-mins	1063	>>	175		155	16800
2-11 Shelf Slots	2562	>	2078		1842	450
2-12 Shelf Slots	1002	>	853		826	236
2-(11+12) Crane-mins	397	>>	98	=	96	6001
3-15 Shelf Slots	2351		2738		2564	79
3-16 Shelf Slots	2493	>	1309		1017	180
3-(15+16) Man-mins	249		175		166	4500

5.5 DISCUSSION

This Chapter extended the theoretical work of the previous chapters to the design of a practical system. We illustrated our methods with details of the system implemented at the FIAT warehouse in Volvera. In addition to the operational aims of the Resource Management task, this system is being used extensively as a design tool, for evaluating possible modifications to the warehouse.

APPENDIX 5A

EXAMPLE OF CALCULATION OF USAGE FUNCTION

This Appendix shows a typical calculation of $u^i(\underline{d}^i, \underline{x}^i)$ for a part, according to the approach of sec.5.2.1. The resource in concern will be the crane-time for a part stored in the "High Shelf Area" and picked at the "bays" (see Fig.2-1 and accompanying description).

Containers for such parts arrive at the High Shelf Area via an input ramp. From there, they are loaded into the shelves by a crane. Picks for these parts are serviced by a crane taking a container down to a manned bay, and after the pick is done, by returning the container to the shelf. When a container is emptied by a pick at the bay, it is taken to an exit ramp, instead of back to the shelf.

We now demonstrate the calculation of total crane-time per day used by such a part. We have kept the calculation simple for illustration purposes; more sophisticated assumptions and calculations could easily be made in the same way.

The part-data $\underline{d}^i$ required for this calculation can be summarized in:

C = no. of Containers emptied per day.

The warehouse operating data required is:

TL = crane Time to Load new containers into shelf from input ramp

TS = crane Time to Supply container to bay from shelf

TR = crane Time to Retrieve container from bay to shelf

TU = crane Time to Unload empty container from bay to exit ramp

(All the above data items are mean values.)

Let [C] = integer part of C

{C} = fractional part of C (= C-[C])

Since partly full containers are replaced on the shelf, a fractional value of C, say 0.25, means that (on average) the container will be bay-picked and replaced on 3 days, and bay-picked and emptied on the fourth. Thus:

Bay Picking Crane Time/day

= C*TL + [C]*(TS+TU) + {C}*(TS+TU) + (1-{C})*(TS+TR)

= C*TL + C*(TS+TU) + (1-{C})*(TS+TR) .

(loads) (empties) (partly-fulls)

CHAPTER 6

APPLICATION TO THE FILE ALLOCATION PROBLEM

6.1 INTRODUCTION

6.1.1 Overview Of Chapter

With the growing size of distributed computer networks, the problem of allocation of resources in these networks is becoming increasingly important. This Chapter addresses one specific aspect of the problem: that of File Allocation. When some information (say, a "file") is shared by several sites ("nodes"), the question arises as to where copies of the file should reside. The decision for a particular file may depend on its Query and Update traffic at each node, on the Storage costs, the Transmission costs, on Response-time constraints, and Reliability-level constraints. Thus the problem for a single file is itself quite complex.

Now, when several thousand files are to be allocated in a network, in addition to minimizing the above costs, the network constraints (storage capacities, link transmission capacities) have also to be taken into account. In this case the problem becomes extremely complex; in fact,

solutions in the literature have (typically) addressed problems with less than 20 files (see sec.6.1.4).

In this Chapter we show how, by appropriate formulation of the problem, a decentralized approach is possible. This brings previously intractable problems (with several thousand files, say) within the realm of known solution methods. Our aim is to show the applicability and advantages of the decentralized approach. For the general optimization problem, we do not go into details of solution algorithms, but give a framework for future research. We do however, consider one case in detail -- the problem faced by a "Network Manager" whose task is to keep a given network operational (that is, all resource usages within the constraints) in the face of constant arrivals of new files, and changing characteristics of old files.

6.1.2 Problem Statement

The general problem is to allocate a number of files to different nodes in a computer network, so as to minimize the total storage costs (of the files) and communication costs (of queries and updates), subject to various constraints. The cost tradeoff can be illustrated by a simple argument for one file. At one extreme, if a copy is stored at each node, then queries at any node can be rapidly answered, without any transmission costs. However, in this case an update at any node has to be transmitted to all the other nodes. Hence, the storage and update-transmission costs will be high for this case. At the other extreme, if the file is stored at only one node, then queries arising at all other nodes need to be transmitted to this node, and in this case the query cost is high, but the storage and update costs are low. (The response time for queries will also be large.) This argument is made precise in the following description.

Basic Parameters (Given)

Set of Files	$f \in \{1,\ldots,F\}$	
Set of Nodes	$n \in \{1,\ldots,N\}$	
Set of Links	$k \in \{1,\ldots,K\}$	Unidirectional links which connect certain nodes according to (given) topology
File Length Vector	$\underline{b}$	b_f = length (in bits) of file f.
Query Traffic Matrix	Q	Q_{fn} = transmission traffic (bits/sec) generated at node n, for queries to file f.
Update Traffic Matrix	U	U_{fn} = analogous to Q_{fn}, but for updates to f.
Node Reliability Vector	$\underline{r}$	r_n = reliability of node n (probability that node n is functioning at any time)
Link Reliability Vector	$\underline{r}$	r_k = reliability of link k
Link Capacity Vector	$\underline{h}$	h_k = capacity of link k (bits/sec)

(Parameters relating to cost are described under a separate heading.)

Decision Variables (All zero-one variables)

These are defined formally below. X is the decision as to how many copies of each file are to be stored, and where. Y is the decision as to how updates originating at a particular node, and for a particular file, will be routed (to all copies of the file). Z is the similar decision for queries (which need only be routed to one copy of the file).

File Allocation Matrix	X	X_{fn} = 1 if file f is allocated to node n.
Update Path Allocation Array	Y	Y_{fknm} = 1 if updates for file f, originating at node n and destined for node m, will use link k.
Query Path Allocation Array	Z	Z_{fkn} = 1 if queries for file f, originating at node n, will use link k on their way to some copy of the file.

Constraints

File Availability Vector	$\underline{a}$	a_f = minimum probability that at least one copy of file f is accessible at any time (depends on node and link reliabilities).
Storage Capacity Vector	$\underline{s}$	s_n = on-line storage capacity (bits) at node n.
Traffic Intensity Vector	$\underline{p}$	traffic (bits/sec) over link k must not exceed $p_k h_k$.

(The availability constraint also ensures that at least one copy of each file is allocated.) The significance of certain terms above will be discussed further in sec.6.1.3.

Costs and Objective Function

Storage Cost Vector	$\underline{c}^S$	c_n^S = cost of storing one bit for one second at node n.
Transmission Cost Vector	$\underline{c}^T$	c_k^T = cost of transmitting one bit over link k.

The objective is to find {X,Y,Z} to minimize:

Total Storage Costs (function of X)
+ Total Transmission Costs (function of X, Y, Z)

subject to the constraints on:

File Availability (function of X)
Storage Capacity (function of X)
Traffic Intensity (function of X, Y, Z)

A mathematical formulation of the above will be given later.

6.1.3 Comments And Comparison With Other Models

In the literature on the File Allocation Problem, although the flavour of the problem is the same, when we investigate details we find no generally accepted model. Retrieval time, which is given so much emphasis in [C2] and [M1] is completely missing in [C3] and [L7]. Similarly, File Availability is considered only in [M1], and Storage Capacity appears only in [C2]. Thus there appears to be a wide range of acceptable models in this field. In comparison with other work, our model has the following features:

1. File Availability. This rather complex constraint, which leads to highly nonlinear constraint equations (see analysis in [M1]), is retained in our model, and we shall see that it is easily incorporated in our analysis.

2. Link Capacity. In some formulations [M1] choice of link capacity is also included as a decision variable in the model. We assume an existing network with given capacities.

3. Retrieval Time. Instead of using this constraint directly, as in [C2] or [M1], we have a traffic intensity (or congestion) constraint for each link. The justification for this is that the average retrieval times between nodes are a function of the

traffic intensities (see [M1]), and by specifying the latter, we can calculate upper bounds for the retrieval times.

4. Decisions. We are able to incorporate even the (very high-dimensional) arrays Y and Z in our formulation, and to solve for them quite easily.

5. Program Allocation. In some formulations (e.g. [L7]) the file-allocation problem is considered simultaneously with the program allocation problem. We do not consider program allocation at all.

The approach taken by Chang [C1] is of a different nature. He considers an extremely general model of the Distributed Computer System Design Problem, with a very large number of parameters. His goal, however, is to find a system configuration satisfying various performance requirements and design constraints. The criterion of optimality is thus replaced by user satisfaction with a design.

A major criticism of all the existing models has been given by Rothnie and Goodman [R4]. They state that these models neglect three important factors:

1. The models do not adequately reflect user demand for database access involving more than one file at the same time.

2. The models completely neglect the synchronization costs involved in updating redundantly stored data. This cost varies in a complex way with the other design decisions.

3. The models assume that complete files should be the unit of assignment to nodes. (There are situations in which an optimal partition of a file will give better results.)

Their criticism is well-taken, and our model also suffers from these shortcomings. However, we feel that even the existing models have not yet been solved efficiently. Our efforts are a step in the direction of solving such models, and only after this can we hope to solve the even more complex ones obtained by including the suggested features.

6.1.4 Comments On Existing Solution Methods

A cross-section of recent work in this area is represented by [C2], [C3], [L7], [M1], [W1]. A review can be found in [L7]. The solution methods vary from integer programming [C2] and linear programming [W1], to a heuristical technique combining dynamic programming and search methods [M1]. All methods suffer from intractability as soon as the problem becomes large. Typical problems solved are given in Table 6-I, where we see that a problem with 5 files and 20 nodes would be of the order of the

largest solved, and would take 16 minutes of CPU time. The area where we intend to research this problem -- 1,000 to 100,000 files and 10 to 20 nodes -- is clearly out of range of these methods.

Another shortcoming of the solution methods relates to a problem encountered in everyday systems: how to allocate a new file that has appeared in the system? The methods offer no simple way of making this decision, short of re-solving the whole problem for the new set of files. Again, this is an area where we can make a contribution.

A useful method from our point of view, and one we shall return to in sec.6.2.3, is that of Casey [C3]. He solves the allocation problem for copies of one file only, with storage and communication costs, but no constraints. Since we shall refer to his results, we summarize the problem solved by Casey in the next section.

Table 6-I: Typical File-Allocation Problems Solved

Reference	Problem Size	CPU Time	Computer Used
[C2] Chu	5 Files 3 Nodes Only 1 copy of each file	25 sec	IBM 360/65
[C3] Casey	1 File 19 Nodes	2 sec	IBM 360/91
[M1] Mahmoud & Riordan	95 (Files x Nodes) 170 (Links x Capacities)	16 min	XDS Sigma 7

Notes:

1. It is not possible to make an even comparison, since the assumptions used differ so much in each case above. For example, [C2] and [M1] used complex constraints, while [C3] had no constraints. [C2] did not allow copies of files, while the others did. [M1] included decisions on link-capacities. Thus the above Table should not be used to compare the different solutions, but only to get an idea of the computer times involved.

2. Casey's numbers were for a total of 6 runs, so our figure above is an average of those runs. Also, Casey solved for both a "linear" and a "nonlinear" relaying assumption for updates; the figure above is for the linear case.

6.1.5 Casey's "One-File" Problem

To avoid duplication of notation, we shall describe Casey's problem for one file using our notation of sec.6.1.2, but with the value of the subscript f fixed, that is, a specified file. The given parameters are Query and Update traffic (Q_{fn} and U_{fn}, for each n), file length (b_f), and storage costs ($\underline{c}^S$). Instead of deciding on query paths and update paths as above, Casey assumes simply that a cost of unit transmission from node n to node $\bar{n}$ is given ($=d_{n\bar{n}}$), and there are no link capacity constraints. The decision to be made is only the file allocation (X_{fn}, for each n), that is, at which node should copies of f reside ? For a given allocation, say $X_{fn}=1$ for $n \in N^f$, the total cost incurred is a function of N^f:

$$[6.1.1] \quad C(N^f)=\sum_{n=1}^{N}\left[U_{fn}\sum_{\bar{n}=1}^{N}X_{f\bar{n}}d_{n\bar{n}}+ Q_{fn}\min_{\bar{n}\in N^f} d_{n\bar{n}}+ b_f c_n^S X_{fn}\right]$$

This can be seen as follows -- for a given node, say n, all updates arriving at n must be transmitted to all copies of n (see first term above), queries will be answered from the "cheapest" node (second term), and there will be a storage cost (third term). $C(N^f)$ is now obtained by summing over n.

Casey's solution method is best visualized using the hypercube of all possible allocations for file f. Fig.6-1 illustrates this for N=4 nodes. Each vertex of the hypercube represents a value of the N-dimensional binary vector $(X_{f1},X_{f2},\ldots,X_{fN})$. Each downward edge joining two

vertices represents an addition of exactly one copy of the file. Let us consider all possible downward paths, from (0,0,...,0) to (1,1,...,1). A theorem by Casey states that along any such patn, whenever a cost increase is encountered in a step from a vertex V to the next vertex, that path can be abandoned, since it will not lead to any points lower in cost than V. Using this property, a "path tracing" algorithm of Casey can be used to find the optimal allocation, without computing the cost of every vertex.

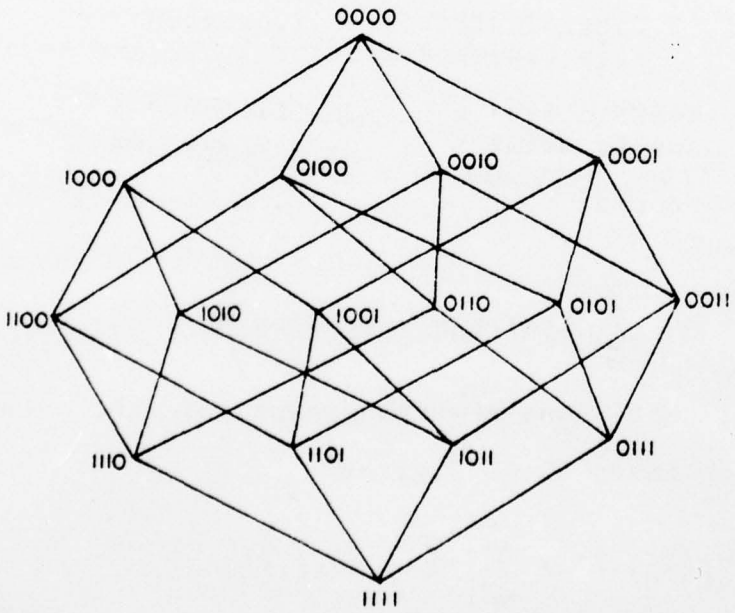


Fig.6-1: Hypercube of Allocations for 4 Nodes

6.2 ANALYSIS OF MODEL

6.2.1 Mathematical Formulation Of Problem

[6.2.1] Transmission Costs

Let $TC_f(X,Y,Z)$ ≙ Total Transmission costs for file f when decisions X,Y,Z are taken

$$= \sum_{n=1}^{N} \left\langle \begin{array}{c} \text{Cost of Updates} \\ \text{from n to all} \\ \text{copies of f} \end{array} + \begin{array}{c} \text{Cost of Queries} \\ \text{from n to the} \\ \text{specified copy} \\ \text{of f} \end{array} \right\rangle$$

$$= \sum_{n=1}^{N} \left\langle U_{fn} \sum_{\bar{n}=1}^{N} \left(X_{f\bar{n}} \sum_{k=1}^{K} c_k^T Y_{fkn\bar{n}} \right) + Q_{fn} \sum_{k=1}^{K} c_k^T Z_{fkn} \right\rangle$$

[6.2.2] Traffic over Links

Let $T_{fk}(X,Y,Z)$ ≙ Total Traffic over link k due to file f when decisions X,Y,Z are taken

$$= \sum_{n=1}^{N} \left\langle \begin{array}{c} \text{Traffic over k} \\ \text{due to Updates} \\ \text{from n to all} \\ \text{copies of f} \end{array} + \begin{array}{c} \text{Traffic over k} \\ \text{due to Queries} \\ \text{from n for f} \end{array} \right\rangle$$

$$= \sum_{n=1}^{N} \left\langle U_{fn} \sum_{\bar{n}=1}^{N} X_{f\bar{n}} Y_{fkn\bar{n}} + Q_{fn} Z_{fkn} \right\rangle$$

[6.2.3] Remark: Note that re-arrangement of the equation for transmission costs [6.2.1] gives

$$[6.2.4] \qquad TC_f(X,Y,Z) = \sum_{k=1}^{K} c_k^T T_{fk}(X,Y,Z)$$

as is to be expected, since the total transmission costs can be thought of as arising in either of these two ways.

[6.2.5] Problem Statement

Define the set W of admissable decisions to consist of decisions (X,Y,Z) which satisfy the following:

X is such that each file f satisfies its availability constraint.

Y is such that for each f, the set of links $\{k \mid Y_{fkn\bar{n}}=1\}$ form a path from node n to $\bar{n}$.

Z is such that for each f, the set of links $\{k \mid Z_{fkn}=1\}$ form a path from node n to some node $\bar{n}$ with $X_{f\bar{n}}=1$.

The problem can then be stated as

$$[6.2.6] \quad \min_{(X,Y,Z)\in W} \sum_{f=1}^{F} \sum_{n=1}^{N} c_n^S b_f X_{fn} + \sum_{f=1}^{F} TC_f(X,Y,Z)$$

(storage costs) + (transmission costs)

subject to

$$\sum_{f=1}^{F} b_f X_{fn} \leq s_n \quad \text{for each } n \quad \text{(storage capacity)}$$

$$\sum_{f=1}^{F} T_{fk}(X,Y,Z) \leq p_k h_k \quad \text{for each } k \quad \text{(link capacity)}$$

6.2.2 Lagrange Multiplier Approach

For the problem above, we use conventional optimization theory [B1], [L5], [L1], to formulate the Lagrangian. Let

$\underline{\Pi}$ = Multiplier for Traffic Constraints

$\underline{\nabla}$ = Multiplier for Storage Constraints

The Lagrangian L is then given by

$$L = \sum_{f=1}^{F} \sum_{n=1}^{N} c_n^S b_f X_{fn} + \sum_{f=1}^{F} TC_f(X,Y,Z)$$

$$+ \sum_{n=1}^{N} \bar{v}_n \left(\sum_{f=1}^{F} b_f X_{fn} - s_n \right)$$

$$+ \sum_{k=1}^{K} \bar{u}_k \left(\sum_{f=1}^{F} T_{fk}(X,Y,Z) - p_k h_k \right)$$

Using [6.2.4] and also defining the modified "costs"

$$\underline{u} \hat{=} \bar{\underline{u}} + \underline{c}^T$$

$$\underline{v} \hat{=} \bar{\underline{v}} + \underline{c}^S$$

we rearrange L to get

$$L(X,Y,Z,\underline{u},\underline{v}) = \sum_{f=1}^{F} \left\langle b_f \sum_{n=1}^{N} v_n X_{fn} + \sum_{k=1}^{K} u_k T_{fk}(X,Y,Z) \right\rangle$$

$$- \sum_{n=1}^{N} v_n s_n - \sum_{k=1}^{K} u_k p_k h_k + \text{constants}$$

The saddle-point theorem [L1] states that a solution to the original problem [6.2.6] will be obtained if we can find $(X^*,Y^*,Z^*,\underline{u}^*,\underline{v}^*)$ which simultaneously satisfy:

$$[6.2.7] \qquad (X^*,Y^*,Z^*) = \arg \min_{(X,Y,Z) \in W} L(X,Y,Z,\underline{u}^*,\underline{v}^*)$$

$$[6.2.8a] \qquad \underline{u}^* = \arg \max_{\underline{u} \geq \underline{c}^T} L(X^*,Y^*,Z^*,\underline{u},\underline{v}^*)$$

$$[6.2.8b] \qquad \underline{v}^* = \arg \max_{\underline{v} \geq \underline{c}^S} L(X^*,Y^*,Z^*,\underline{u}^*,\underline{v})$$

where the vector inequalities are to be interpreted componentwise.

6.2.3 Decentralized Solution Technique

This approach has already been described in Chapters 2 and 3, and is suggested by the statements [6.2.7] and [6.2.8] above. The decentralized solution algorithm starts with initial guesses for $\underline{u}$ and $\underline{v}$, and solves [6.2.7] to get $X^{(1)}$, $Y^{(1)}$, and $Z^{(1)}$, say. It then uses these values of X, Y, Z, to modify $\underline{u}$ and $\underline{v}$ slightly, so that improvements in the objectives in [6.2.8] are obtained. This process is repeated until equilibrium is obtained, at which point the values of X, Y, Z, will be solutions to the optimization problem [6.2.6]. Details of this approach, as well as conditions for its convergence, can be found in [A1]. Our aim here is to examine its applicability, which will be brought out in the Theorem below.

[6.2.9] Theorem 6-I (Decomposition of File-Allocation Problem)

The optimization problem (for given $\underline{u}$, $\underline{v}$):

[6.2.10] Find $(X^*,Y^*,Z^*) = \arg \min_{(X,Y,Z)\in W} L(X,Y,Z,\underline{u},\underline{v})$

is equivalent to a set of F independent individual file-allocation problems, and each of these individual problems is equivalent to the problem solved by Casey (sec.6.1.5). []

Proof: We will omit terms not containing X, Y, or Z. Also, for notational convenience we define the following vectors:

$$\text{[6.2.11]} \qquad \begin{aligned} &\text{vector } \underline{x}^{f} \quad \text{such that } x_n^f = X_{fn} \\ &\text{vector } \underline{y}^{fn\bar{n}} \quad \text{such that } y_k^{fn\bar{n}} = Y_{fkn\bar{n}} \\ &\text{vector } \underline{z}^{fn} \quad \text{such that } z_k^{fn} = Z_{fkn} \end{aligned}$$

In the above, if we restrict (X,Y,Z) to range over W, then the values of $\underline{x}^f$, $\underline{y}^{fn\bar{n}}$, $\underline{z}^{fn}$, generate the corresponding sets of admissable decision vectors. These sets will be denoted by X^f, $Y^{fn\bar{n}}$, Z^{fn} respectively. Also define

$d_{n\bar{n}}$ = cost of minimum-cost path from node n to $\bar{n}$ (for given transmission cost vector $\underline{u}$).

The problem is

$$\min_{(X,Y,Z)\in W} \sum_{f=1}^{F} \left\langle \sum_{k=1}^{K} u_k T_{fk}(X,Y,Z) + b_f \sum_{n=1}^{N} v_n X_{fn} \right\rangle$$

which, using [6.2.1]-[6.2.4] becomes

$$\min_{(X,Y,Z)\in W} \sum_{f=1}^{F} \sum_{n=1}^{N} \left\langle U_{fn} \sum_{\bar{n}=1}^{N} X_{f\bar{n}} \sum_{k=1}^{K} u_k Y_{fkn\bar{n}} + Q_{fn} \sum_{k=1}^{K} u_k Z_{fkn} + b_f v_n X_{fn} \right\rangle$$

Now, by considering the independence of certain decisions (since there are no constraints for this problem), we can interchange some of the "min" and "$\sum$" operations, to get:

$$\sum_{f=1}^{F} \min_{\underline{x}^f \in X^f} \sum_{n=1}^{N} \left\langle U_{fn} \sum_{\bar{n}=1}^{N} x_{\bar{n}}^f \min_{\underline{y}^{fn\bar{n}} \in Y^{fn\bar{n}}} \left(\sum_{k=1}^{K} u_k y_k^{fn\bar{n}} \right) \right.$$

$$\left. + Q_{fn} \min_{\underline{z}^{fn} \in Z^{fn}} \left(\sum_{k=1}^{K} u_k z_k^{fn} \right) + b_f v_n x_n^f \right\rangle$$

Defining the set of nodes $N^f=\{n \mid x_n^f=1\}$ and using the definition of $d_{n\tilde{n}}$ above, this equals

$$\sum_{f=1}^{F} \min_{\underline{x}^f \in X^f} \sum_{n=1}^{N} \left\{ U_{fn} \sum_{\tilde{n}=1}^{N} x_{\tilde{n}}^f d_{n\tilde{n}} + Q_{fn} \min_{\tilde{n} \in N^f} d_{n\tilde{n}} + b_f v_n x_n^f \right\}$$

which, for each file f, is <u>exactly the file allocation problem solved by Casey</u> [6.1.1]. (The <u>availability constraint</u> $\underline{x}^f \in X^f$ can be added on with ease. We simply constrain Casey's algorithm to start from the set of highest vertices in the hypercube, which satisfy the availability criterion, i.e. are members of X^f. For a calculation of availability, see [M1]. Note that the above set of vertices need only be calculated once, for each file.)

Thus the optimal decisions are given by:

$$\hat{\underline{x}}^f = \arg \min_{\underline{x}^f \in X^f} \sum_{n=1}^{N} \left\{ U_{fn} \sum_{\tilde{n}=1}^{N} x_{\tilde{n}}^f d_{n\tilde{n}} + Q_{fn} \min_{\tilde{n} \in N^f} d_{n\tilde{n}} + b_f v_n x_n^f \right\}$$

[6.2.12]
$$\hat{\underline{y}}^{fn\tilde{n}} = \arg \min_{\underline{y}^{fn\tilde{n}} \in Y^{fn\tilde{n}}} \sum_{k=1}^{K} u_k y_k^{fn\tilde{n}}$$

$$\hat{\underline{z}}^{fn} = \arg \min_{\underline{z}^{fn} \in Z^{fn}} \sum_{k=1}^{K} u_k z_k^{fn}$$

Note that $\hat{\underline{x}}^f$ can be calculated using Casey's algorithm, while $\hat{\underline{y}}^{fn\tilde{n}}$ and $\hat{\underline{z}}^{fn}$ are simply the minimum cost paths between certain nodes, as calculated for $d_{n\tilde{n}}$ above. []

[6.2.13] Interpretation of Theorem

We have thus derived rigorously the appealing result that the problem [6.2.10] can be approached individually for each file. Note that the actual costs are replaced by the artificial costs ($\underline{u}$,$\underline{v}$) when we solve the individual file allocation problem. Conceptually, we have created a "File Manager" for each file, whose job is to allocate his file in the cheapest manner, for given artificial costs $\underline{u}$ and $\underline{v}$. []

[6.2.14] Remark: Morgan and Levin [M2] showed that the multifile minimization problem could be decomposed into individual file minimization problems. Their proof however, was in the absence of any link capacity or storage capacity constraints. Our result is therefore much more significant. []

6.2.4 Conjectured Iteration Algorithm

Analogous to the File Managers solving [6.2.10] for each file, we can think of a Network Manager who is in charge of satisfying the network constraints, and who changes the artificial costs by small amounts so as to get an improvement in the objective functions of [6.2.8]. Inspection of these objectives shows that (in accordance with our intuition) if the usage of a resource is over its limits, its artificial cost will be increased by the Network

Manager, and vice-versa, except that the artificial cost will not be lowered below the actual cost.

[6.2.15] Algorithm (Simple Version)

The simplest decentralized solution algorithm would thus consist of a FM (File-Manager) Subroutine, which would solve [6.2.7] for each file, followed by a NM (Network-Manager) Subroutine, which would change the resource usages as above. The algorithm would iterate as described in sec.6.2.3.

6.3 ALGORITHM FOR THE FEASIBILITY PROBLEM

The iteration method above is rather crude, and may converge extremely slowly, or not at all. In this section, we describe a special case of the File Allocation problem, for which an efficient algorithm exists, and for which the conditions for convergence are, by comparison, considerably less stringent.

Suppose the network has a very large number of files (say, 10,000) and is managed by a network manager whose task is to keep the network operational, i.e. all the resource usages within the constraints. Suppose also that he must find allocations (in real-time) for several newly created files every day. In addition he must periodically re-allocate existing files according to their changing characteristics, and according to the changes in total

resource usages. His tasks can thus be summarized as:

1. Find an initial feasible solution.

2. Find a rationale for allocating new files in real-time (hence must be a simple algorithm).

3. Maintain the resource usages below their constraints by periodic re-allocation.

The main problem above is to find a feasible solution, that is, a set of allocations and query/update path decisions, which satisfy all the storage constraints, link capacity constraints, and file availability constraints. For a large number of files, this in itself could be a major problem. An economic interpretation of this problem would be as follows: Suppose a network has already been constructed with certain storage and link capacities, and the additional cost of using these facilities is negligible in comparison with the original investment. Then the main concern of the Network Manager is one of feasibility. A design-time interpretation of this approach would be to say it is of the type advocated by Chang [C1], as described in sec.6.1.3.

The feasibility problem has already been studied by us in Chapters 2 to 4. Our results are applied below to the file allocation problem. We only outline the application. The details are an easy extension of the work in previous Chapters.

AD-A064 780 HARVARD UNIV CAMBRIDGE MA DIV OF APPLIED SCIENCES F/G 5/1
RESOURCE MANAGEMENT IN LARGE SYSTEMS.(U)
DEC 78 R SURI
UNCLASSIFIED TR-671 N00014-75-C-0648
NL

3 OF 3
AD
A064780

END
DATE
FILMED
4-79
DDC

6.3.1 Application Of Our Results

[6.3.1] Statement of the Feasibility Problem

Find (X,Y,Z) such that

$$(X,Y,Z) \in W \quad \text{(includes availability constraint)}$$

$$\sum_{f=1}^{F} b_f X_{fn} \leq s_n \quad \text{for each } n \quad \text{(storage capacity)}$$

$$\sum_{f=1}^{F} T_{fk}(X,Y,Z) \leq p_k h_k \quad \text{for each } k \quad \text{(link capacity)}$$

[6.3.2] Statement of the Lagrangian Problem

Let $\underline{u} \geq 0$ and $\underline{v} \geq 0$ be given, and let $\underline{\lambda} \triangleq (\underline{u}', \underline{v}')'$. Let $\hat{\underline{x}}^f$, $\hat{\underline{y}}^{fn\bar{n}}$, $\hat{\underline{z}}^{fn}$, be as in [6.2.12]. The Lagrangian Problem is to find the "decentralized solutions" $X(\underline{\lambda})$, $Y(\underline{\lambda})$, $Z(\underline{\lambda})$ where:

$$X(\underline{\lambda}) \text{ is such that } X_{fn} = \hat{x}_n^f$$

$$Y(\underline{\lambda}) \text{ is such that } Y_{fkn\bar{n}} = \hat{y}_k^{fn\bar{n}}$$

$$Z(\underline{\lambda}) \text{ is such that } Z_{fkn} = \hat{z}_k^{fn} .$$

[6.3.3] Existence Theorem

(For a formal statement see Chapter 3). This theorem says that if the problem [6.3.1] is still feasible for a slightly tighter set of storage and link capacity constraints, then there will exist multipliers $\underline{\lambda}^*$ such that the decentralized solutions $X(\underline{\lambda}^*)$, $Y(\underline{\lambda}^*)$, $Z(\underline{\lambda}^*)$, will be feasible for the problem [6.3.1]. []

[6.3.4] Remarks

1. A major drawback to the use of decentralized solutions in general, is the existence of problems for which no decentralized solutions exist -- see the remarks on "duality gaps" in Chapter 2, Sec.7. Our theorem above gives conditions which ensure the existence of a suitable $\underline{\lambda}^*$, and also, the remarks in Chapter 3 show that these conditions are likely to be true for a large system.

2. As pointed out in Theorem 6-I, for given $\underline{\lambda}$, the decentralized solutions are relatively easy to find. The problem [6.3.1] is thus reduced to one of finding an appropriate $\underline{\lambda}^*$. This will be our next topic of discussion.

6.3.2 Finding Suitable Multipliers

In Chapter 4 we gave algorithms for finding multipliers such that the decentralized solutions solved the feasibility problem. Consider the SALA-3 iteration scheme [4.6.11]. The reader can see that a simple modification of this scheme (replace the step "INITMCA" by a solution of [6.2.12], and define the resource usage vector to be the storage and link capacity usages for a given solution) gives us an algorithm for the file allocation problem. Theorem 4-IV in Chapter 4

shows that, under suitable assumptions, such an iteration scheme converges to a solution in a finite number of iterations, at a better than geometric convergence rate. Discussions in Chapter 4 also demonstrate that the assumptions are reasonable for a large system, and this has been upheld by use of the algorithm in practice (Chapter 5).

6.3.3 Conjectured Efficiency Of Iteration Methods

Experiments with the SALA algorithm (see Chapter 5) have shown that it converges in 5 to 15 iterations, irrespective of the number of items (in this case, files). The solution time on average is (say) 10 iteration cycles. The time for each iteration is proportional to

(no. of files) x (time to solve one-file problem) .

Now Casey's algorithm for solving the one-file problem has been shown to be NP-complete [E2], as a function of the number of nodes (N). However, for a small number of nodes (under 20), the time to solve the one-file problem is still reasonable (2 CPU-secs for 19 nodes, see Table 6-I). The solution time for our method is therefore expected to increase linearly with the number of files and exponentially with the number of nodes. The time for Integer Programming solutions would be exponentially proportional to each of these. Thus our approach is expected to be superior for problems with a very large number of files (1000 to 100,000) and a moderate number of nodes (10 to 20).

However, note that since the changes in $\underline{\lambda}$ at each iteration are small, it may be possible to re-solve the one-file problems in a more efficient manner, using the results of the previous iteration. This would extend applicability to problems with a larger number of nodes, and is a subject for further research.

6.3.4 New Files And Periodic Re-allocation

The previous sections considered the (static) file allocation problem. In an operational network, we have the two additional problems, "new file allocation" and "periodic reallocation" (sec.6.3.1). The solution of these problems in an operational system has been discussed in Chapter 5. The reader can see that this approach can be used for the file allocation problem. Briefly, the new file allocation problem is solved by using the current "costs" $\underline{\lambda}$ to solve the one-file problem for a new file (can be done in Real-Time), and the periodic review problem is solved by finding a small change in $\underline{\lambda}$ such that the resulting re-allocation will restore usages to their desired levels. See Chapter 5 for details.

6.4 SUMMARY

We have demonstrated the scope for application of decentralized solution techniques to the file allocation

problem. Our formulation allows the incorporation of several different types of constraints, including the rather complex file availability constraints. We outlined a possible solution algorithm for the general problem. In a special case of this problem, we showed how results from Chapters 3 to 5 could be used directly for solution of the problem. Our methods are efficiently applicable for systems with a very large number of files, and a moderate number of nodes. Further research may extend applicability to problems with a large number of nodes too. The size of problems that we can solve efficiently (10,000 files and 10 nodes, say) would certainly be considered intractable for other existing solution techniques.

CHAPTER 7

CONCLUSIONS: A NEW APPROACH TO LARGE SCALE SYSTEMS?

We have studied the task of Resource Management in Large Systems. This task was defined as having three main objectives: Initial-Assignments, New-Assignments, and Periodic Review. The importance of this task was illustrated by examples from a large warehouse, and from a computer network.

The Resource Management task was formulated as a certain "Feasibility Problem". The decentralized approach was made possible by formulating an equivalent "Artificial" optimization problem, and decomposing it through the use of Lagrange Multipliers. Chapter 3 studied the existence of optimal multipliers for this artificial problem, while Chapter 4 developed iteration algorithms to find these multipliers. Chapter 5 demonstrated the application of our techniques in a practical system, and Chapter 6 illustrated the applicability of our methods to the file allocation problem in computer networks.

The advantage of decentralized techniques is that they make possible the efficient solution of very large problems. However, the applicability of these techniques has been restricted to problems which satisfy strict conditions. The main contribution of our work is to extend the applicability of decentralized solution methods to problems where the resource usage functions are not well-behaved. We give conditions for the existence of decentralized solutions, and also give algorithms to find such solutions, without requiring strict conditions on the functions.

The type of assumptions and conditions required for our results in Chapters 3 and 4, reflect properties of the system as a whole, rather than the properties of the individual items in the system. We feel that this is an important viewpoint for dealing with large systems:

> Results in many areas of Large-Scale Systems theory depend on strict conditions on the functions characterizing the elements of the system, which we might call the local properties of the system. We feel that the structural properties of a large system (which we might call the global properties) may enable us to make statements about the system, without requiring stringent conditions on the local properties. This may greatly extend our ability to apply known techniques, and develop new techniques, for problems in the area of Large Scale Systems.

The advantage of decentralized techniques is that they make possible the efficient solution of very large problems. However, the applicability of these techniques has been restricted to problems which satisfy strict conditions. The main contribution of our work is to extend the applicability of decentralized solution methods to problems where the resource usage functions are not well-behaved. We give conditions for the existence of decentralized solutions, and also give algorithms to find such solutions, without requiring strict conditions on the functions.

The type of assumptions and conditions required for our results in Chapters 3 and 4, reflect properties of the system as a whole, rather than the properties of the individual items in the system. We feel that this is an important viewpoint for dealing with large systems:

> Results in many areas of Large-Scale Systems theory depend on strict conditions on the functions characterizing the elements of the system, which we might call the local properties of the system. We feel that the structural properties of a large system (which we might call the global properties) may enable us to make statements about the system, without requiring stringent conditions on the local properties. This may greatly extend our ability to apply known techniques, and develop new techniques, for problems in the area of Large Scale Systems.

We have already noted, in Chapter 1, the growing importance of Large-Scale systems and Decentralized Control. In this context, two fundamental questions come to mind:

1. When is a system "large" ?

2. When is decentralized control "good enough" ?

The assumptions and results of our work suggest that perhaps these questions should be related by definition, that is, a system should be considered "large" when decentralized control can provide acceptable performance.

REFERENCES

[A1] Arrow, K.J. and Hurwicz, L., "Decentralization and Computation in Resource Allocation", in Essays in Economics and Econometrics, R.W.Pfonts (Ed.) Univ. of North Carolina Press (1960).

[A2] Aho, A.V., Hopcroft, J.E., and Ullman, J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley (1974).

[B1] Bryson, A.E. and Ho, Y.C., Applied Optimal Control, Ginn and Company (1969).

[C1] Chang, S.K., "A Model for Distributed Computer System Design", IEEE Trans. Systems, Man and Cybernetics 5 (May 1976) pp.344-359.

[C2] Chu, W.W., "Optimal File Allocation in a Multiple Computer System", IEEE Trans. Computers (October 1969), pp.885-389.

[C3] Casey, R.G., "Allocation of Copies of Files in an Information Network", Proc. AFIPS 1972, Vol.40, pp.617-625.

[D1] Dantzig, G.B., Linear Programming and Extensions, Princeton: Princeton Univ. (1963).

[E1] Everett, H., "Generalized Lagrange Multiplier Method for solving problems of Optimum Allocation of Resources", Operations Research 11 (1963) pp.399-417.

[E2] Eswaran, K.P., "Placement of records in a file and file allocation in a computer network", IFIP Conf. Proc., Stockholm, Sweden (Aug.1974) pp.304-307.

[E3] Ermol'ev, Yu.M. and Shor, N.Z., "On the Minimization of Nondifferentiable Functions", Kibernetika, No.2, 1967.

[F1] FIAT, Volvera: The car spare parts warehouse, FIAT Information and Advertising, Edition No.4398, Turin, Italy.

[F2] FIAT/CSDL, Design of New Operating Procedures for the FIAT Automobile Spare Parts Warehouse at Volvera. Turin, Italy: FIAT Ricambi (April 1977).

[G1] Geoffrion, A.M., "Elements of Large-Scale Mathematical Programming", Management Science 16 (July 1970) pp.652-691. Also in [G2].

[G2] _____, (Ed.) Perspectives on Optimization, Addison-Wesley (1972).

[G3] _____, "Duality in Nonlinear Programming: A Simplified Applications-oriented Development", SIAM Review 13 (Jan.1971) pp.1-37. Also in [G2].

[K1] Katkovnik, V.Ya., "Method of Averaging Operators in Iteration Algorithms for Stochastic Optimization", Cybernetics (USA) 8 (July-Aug.1972) pp.670-679.

[K2] Kushner, H.J., "Convergence of Recursive Adaptive and Identification Procedures via Weak Convergence Theory", IEEE Trans. Aut. Control 22 (Dec.1977) pp.921-930.

[L1] Lasdon, L.S., Optimization Theory for Large Systems, New York: Macmillan (1970).

[L2] Leitmann, G., (Ed.) Multicriteria Decision Making and Differential Games, New York: Plenum (1976).

[L3] Lin, J.G., "Multiple-Objective Optimization: Proper

Equality Constraints (PEC) and Maximization of Index Vectors", in [L2].

[L4] Luenberger, D.G., Optimization by Vector Space Methods, New York: John Wiley (1969).

[L5] _____, Introduction to Linear and Nonlinear Programming, Addison-Wesley (1973).

[L6] Ljung, L., "Analysis of Recursive Stochastic Algorithms", IEEE Trans. Aut. Control 22 (Aug.1977) pp.551-575.

[L7] Levin, K.D. and Morgan, H.L., "Optimizing Distributed Databases -- A Framework for Research", Proc. AFIPS 1975, Vol.45, pp.473-478.

[M1] Mahmoud, S. and Riordan, J.S., "Optimal Allocation of Resources in Distributed Information Networks", ACM Trans. Database Systems 1 (March 1976) pp.66-78.

[M2] Morgan, H.L. and Levin, K.D., Optimal Program and Data Locations in Computer Networks, Tech. Rep. 74-10-01, The Wharton School, Univ. of Pennsylvania (1974).

[N1] Nering, E.D., Linear Algebra and Matrix Theory, 2nd

ed., New York: John Wiley (1970).

[N2] Nikaido, H., Convex Structures and Economic Theory, New York: Academic Press (1968).

[R1] Robinson, S.M., "Extension of Newton's Method to Nonlinear Functions with values in a Cone", Numer. Math. 19 (1972) pp.341-347.

[R2] _____, "Normed Convex Processes", Trans. Amer. Math. Soc. 174 (Dec.1972) pp.127-140.

[R3] Rockafellar, R.T., Monotone Processes of Convex and Concave Type, Princeton: Amer. Math. Soc. Memoirs No. 77 (1967).

[R4] Rothnie, J.B. and Goodman, N. "A Survey of Research and Development in Distributed Database Management", Proc. Conf. VLDB (1977).

[S1] Stoer, J. and Witzgall, C. Convexity and Optimization in Finite Dimensions Vol.I, Berlin: Springer-Verlag (1970).

[S2] Shapiro, J.F., A Survey of Lagrangean Techniques for Discrete Optimization, Tech. Rep. 133, Operations Research Center, M.I.T., Cambridge, MA. (May 1977).

[S3] Suri, R., SALA Reference Manual and User's Guide, Report FR72400-03, C.S.Draper Lab., Cambridge, MA. (Dec.1977).

[S4] Suri, R., SALA: Overflow Level Control, Memo FMT72400-66, C.S.Draper Lab., Cambridge, MA. (June 1977).

[W1] Whitney, V.K.M., A Study of Optimal File Assignment and Communication Network Configuration, SEL Tech. Rep. 42, Univ. of Michigan (Sep.1970).

[Z1] Zangwill, W.I., Nonlinear Programming: A Unified Approach, Prentice-Hall (1969).

[Z2] Zukhovitskiy, S.I. and Avdeyeva, L.I., Linear and Convex Programming, Philadelphia: W.B. Saunders (1966).

JSEP REPORTS DISTRIBUTION LIST

Director
National Security Agency
ATTN: Dr. T. J. Beahn
Fort George G. Meade, MD 20755

Defense Documentation Center
ATTN: DDC-TCA (Mrs. V. Caponio)
Cameron Station
Alexandria, VA 22314

Asst. Dir.
Office of Director of Defense
Research and Engineering
The Pentagon
Washington, DC 20315

Defense Advanced Research
Projects Agency
ATTN: (Dr. R. Reynolds)
1400 Wilson Boulevard
Arlington, VA 22209

Commandant
US Army Air Defense School
ATTN: ATSAD-T-CSM
Fort Bliss, TX 79916

Commander
US Army Armament R&D Command
ATTN: DRSAR-RD
Dover, NJ 07801

Commander
US Army Ballistic Research Lab.
ATTN: DRXBR-BAD
Aberdeen Proving Ground
Aberdeen, MD 21005

Commandant
US Army Command and General
Staff College
ATTN: Acquisitions, Lib. Div.
Fort Leavenworth, KS 66027

Commander
US Army Communication Command
ATTN: CC-OPS-PD
Fort Huachuca, AZ 85613

Commander
US Army Materials and Mechanics
Research Center
ATTN: Chief, Materials Sciences
Division
Watertown, MA 02172

Commander
US Army Materiel Development
and Readiness Command
ATTN: Technical Library, Rm.7S 35
5001 Eisenhower Avenue
Alexandria, VA 22333

Commander
US Army Missile R&D Command
ATTN: Chief, Document Section
Redstone Arsenal, AL 35809

Commander
US Army Satellite Communications
Agency
Fort Monmouth, NJ 07703

Commander
US Army Security Agency
ATTN: IAMG-T
Arlington Hall Station
Arlington, VA 22212

Project Manager
AFTADS
EAT Building
West Long Branch, NJ 07764

Commander/Director
Atmospheric Sciences Lab. (ECOM)
ATTN: DRSEL-BL-DD
White Sands Missile Range, NM 88002

Commander
US Army Electronics Command
ATTN: DRSEL-RL-O (Dr. H.S. Bennett)
Fort Monmouth, NJ 07703

Director
TRI-TAC
ATTN: TT-AD (Mrs. Briller)
Fort Monmouth, NJ 07703

Commander
US Army Electronics Command
ATTN: DRSEL-CT-L (Dr. P. Buser)
Fort Monmouth, NJ 07703

Director
Electronic Warfare Lab. (ECOM)
ATTN: DRSEL-WL-MY
White Sands Missile Range, NM 88002

Executive Secretary, TAC/JSEP
US Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709

Commander
Frankford Arsenal
Deputy Director,
Pitman-Dunn Laboratory
Philadelphia, PA 19137

Project Manager
Ballistic Missile Defense Program Office
ATTN: DACS-SMP (Dr. A. Gold)
1300 Wilson Blvd.
Arlington, VA 22209

Commander
Harry Diamond Laboratories
ATTN: Mr. John E. Rosenberg
2800 Powder Mill Road
Adelphi, MD 20783

HQDA (DAMA-ARZ-A)
Washington, DC 20310

Commander
US Army Electronics Command
Fort Monmouth, NJ 07703
ATTN: DRSEL-TL-E (Dr. J.A. Kohn)
DRSEL-TL-EM (Dr. S. Kronenberg)
DRSEL-ML-T (Mr. R. Kulicke)
DRSEL-ML-B (Dr. E. Liebleim)
DRSEL-TL-MM (Mr. M. Lipetz)
DRSEL-RD-G (Dr. W.S. McAfee)
DRSEL-ML-B (Dr. D.C. Pearce)
DRSEL-ML-MS-1 (Dr. F. Schwering)
DRSEL-TL-I (Dr. C.G. Thornton)

COL Robert Noce
Senior Standardization Representative,
US Army Standardization Group, Canada,
Canadian Force Headquarters
Ottawa, Ontario, Canada K1A 0K2

Director
Night Vision Laboratory
ATTN: DRSEL-NV-O
Fort Belvoir, VA 22060

Commander
Picatinny Arsenal
ATTN: SMUFA-TS-T-S
Dover, NJ 07801

Dr. Sidney Ross
Technical Director
SARFA-TD
Frankford Arsenal
Philadelphia, PA 19137

US Army Research Office (Icpa)
ATTN: Library
P.O. Box 12211
Research Triangle Park, NC 27709

Director
Division of Neuropsychiatry
Walter Reed Army Institute of
Research
Washington, DC 20012

Commander
White Sands Missile Range
ATTN: STEWS-ID-R
White Sands Missile Range, NM 88002

Mr. Robert Barrett
RADC/ETS
Hanscom AFB, MA 01731

Dr. Carl E. Baum
AFWL (ES)
Kirtland AFB, NM 87117

Dr. E. Champagne
AFAL/DH
Wright-Patterson AFB, OH 45433

Dr. R. P. Dolan
RADC/ETSD
Hanscom AFB, MA 01731

Mr. W. Edwards
AFAL/TE
Wright-Patterson AFB, OH 45433

Professor R.E. Fontana
Head, Dept. of Electrical Eng.
AFIT/ENE
Wright-Patterson AFB, OH 45433

Dr. Alan Garscadden
AFAPL/POD
Wright-Patterson AFB, OH 45433

USAF European Office of Aerospace
Research
ATTN: Major J. Gorrell
Box 14, FPO, New York 09510

LTC Richard J. Gowen
Dept. of Electrical Engineering
USAF Academy, CO 80840

Mr. Murray Kesselman (ISCA)
Rome Air Development Center
Griffiss AFB, NY 13441

Dr. G. Knausenberger
Air Force Member, TAC
Air Force Office of Scientific
Research
(AFSC) AFSOR/NE
Bolling Air Force Base, DC 20332

Dr. L. Kravitz
Air Force Member, TAC
Air Force Office of Scientific
Research
(AFSC) AFSOR/NE
Bolling Air Force Base, DC 20332

Mr. R. D. Larson
AFAL/DHR
Wright-Patterson AFB, OH 45433

Dr. Richard B. Mack
RADC/ETER
Hanscom AFB, MA 01731

Mr. John Mottsmith (MCIT)
HQ ESD (AFSC)
Hanscom AFB, MA 01731

Dr. Richard Picard
RADC/ETSL
Hanscom AFB, MA 01731

Dr. J. Ryles
Chief Scientist
AFAL/CA
Wright-Patterson AFB, OH 45433

Dr. Allen Schell
RADC/ETE
Hanscom AFB, MA 01731

Mr. H.E. Webb, Jr. (ISCF)
Rome Air Development Center
Griffiss AFB, NY 13441

LTC G. Wepfer
Air Force Office of Scientific
Research
(AFSC) AFSOR/NP
Bolling Air Force Base, DC 20332

LTC G. McKemie
Air Force Office of Scientific
Research
(AFSC) AFSOR/NM
Bolling Air Force Base, DC 20332

Dr. R. S. Allgaier
Naval Surface Weapons Center
Code WR-303
White Oak
Silver Spring, MD 20910

Naval Weapons Center
ATTN: Code 5515 - H.F. Blazek
China Lake, CA 93555

Dr. H. L. Blood
Technical Director
Naval Undersea Center
San Diego, CA 95152

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375
ATTN: Code 5200 - A. Brodzinsky
Code 7701 - J. D. Brown
Code 5210 - J. E. Davey
Code 5460/5410 - J.R. Davis
Code 5510 - W. L. Faust
Code 2627 - Mrs. O. Folen

Naval Ocean Systems Center
ATTN: Code 75 - W.J. Dejka
271 Catalina Boulevard
San Diego, CA 92152

Naval Weapons Center
ATTN: Code 601 - F.C. Essig
China Lake, CA 93555

Dr. Robert H. Fossum
Dean of Research
Naval Postgraduate School
Monterey, CA 93940

Dr. G. G. Gould
Technical Director
Naval Coastal System Lab.
Panama City, FL 32401

Naval Ocean Systems Center
ATTN: Code 7203 - V.E. Hildebrand
271 Catalina Boulevard
San Diego, CA 92152

Naval Ocean Systems Center
ATTN: Code 753 - P.R. Johnson
217 Catalina Boulevard
San Diego, CA 92152

Donald E. Kirk
Professor and Chairman
Electronic Engineer, SP-304
Naval Postgraduate School
Monterey, CA 93940

Naval Air Development Center
ATTN: Code 01 - Dr. F. K. Lobb
Johnsville
Warminster, PA 18974

Naval Research Laboratory
ATTN: Code 5270 - B.D. McCombe
4555 Overlook Avenue, SW
Washington, DC 20375

Capt. R.B. Meeks
Naval Sea Systems Command
MC 81
2531 Jefferson Davis Hwy
Arlington, VA 20362

Dr. H. J. Mueller
Naval Air Systems Command
Code 310
JP 81
1411 Jefferson Davis Hwy
Arlington, VA 20360

Dr. J.H. Nills, Jr.
Naval Surface Weapons Center
Electronics Systems Department
Code DF
Dahlgren, VA 22448

Naval Ocean Systems Center
ATTN: Code 702 - R.T. Mortimer
271 Catalina Boulevard
San Diego, CA 92152

Naval Air Development Center
ATTN: Technical Library
Johnsville
Warminster, PA 18974

Naval Ocean Systems Center
ATTN: Technical Library
271 Catalina Boulevard
San Diego, CA 92152

Naval Research Laboratory
Underwater Sound Reference Division
Technical Library
P.O. Box 8337
Orlando, FL 32806

Naval Surface Weapons Center
ATTN: Technical Library
Code DX-21
Dahlgren, VA 22448

Naval Surface Weapons Center
ATTN: Technical Library
Building 1-330, Code WR-40
White Oak
Silver Spring, MD 20910

Naval Training Equipment Center
ATTN: Technical Library
Orlando, FL 32813

Naval Undersea Center
ATTN: Technical Library
San Diego, CA 92152

Naval Underwater Systems Center
ATTN: Technical Library
Newport, RI 02840

Office of Naval Research
Electronic & Solid State Sciences
Program (Code 427)
800 North Quincy Street
Arlington, VA 2217

Office of Naval Research
Mathematics Program (Code 432)
800 North Quincy Street
Arlington, VA 22217

Office of Naval Research
Naval Systems Division
Code 220/221
800 North Quincy Street
Arlington, VA 22217

Director
Office of Naval REsearch
New York Area Office
715 Broadway, 5th Floor
New York, NY 10003

Office of Naval Research
San Francisco Area Office
One Hallidie Plaza - Suite 601
San Francisco, CA 94102

Director
Office of Naval Research Branch
Office
495 Summer Street
Boston, MA 02210

Director
Office of Naval Research Branch
Office
536 South Clark Street
Chicago, IL 60605

Director
Office of Naval Research Branch
Office
1030 East Green Street
Pasadena, CA 91101

Mr. H. R. Riedl
Naval Surface Weapons Center
Code WR-34
White Oak Laboratory
Silver Spring, MD 20910

Naval Air Development Center
ATTN: Code 202 - T. J. Shoppie
Johnsville
Warminster, PA 18974

Naval Research Laboratory
ATTN: Code 5403-J. E. Shore
4555 Overlook Avenue, SW
Washington, DC 20375

A. L. Slafkovsky
Scientific Advisor
Headquarters Marine Corps
MC-RD-1
Arlington Annex
Washington, DC 20380

Harris B. Stone
Office of Research, Development,
Test and Evaluation
NOP-987
The Pentagon, Room 5D760
Washington, DC 20350

Mr. L. Sumney
Naval Electronics Systems Command
Code 5j43, MC 81
2511 Jefferson Davis Hwy.
Arlington, VA 20360

David W. Taylor
Naval Ship Research and
Development Center
Code 522.1
Bethesda, MD 20084

Naval Research Laboratory
ATTN: Code 4105-Dr. A. Teitler
4555 Overlook Avenue, SW
Washington, DC 20375

Lt Cdr. John Turner
NAVMAT 0343
CP #5, Room 1044
2211 Jefferson Davis Hwy.
Arlington, VA 20360

Naval Ocean Systems Center
ATTN: Code 746-M.E. Wieder
271 Catalina Boulevard
San Diego, CA 92152

Dr. W. A. Von Winkle
Associate Technical Director
for Technology
Naval Underwater Systems Center
New London, CT 06320

Dr. Gernot M. R. Winkler
Director, Time Service
US Naval Observatory
Massachusetts Avenue at
34th Street, NW
Washington, DC 20390

Dr. Howard W. Etzel
Deputy Director
Division of Materials Research
National Science Foundation
1800 G Street
Washington, DC 20550

Mr. J. C. French
National Bureau of Standards
Electronics Technology Division
Washington, DC 20234

Dr. Jay Harris
Program Director
Devices and Waves Program
National Science Foundation
1800 G Street
Washington, DC 20550

Los Alamos Scientific Laboratory
ATTN: Reports Library
P.O. Box 1663
Los Alamos, NM 87544

Dr. Dean Mitchell
Program Director
Solid-State Physics
Division of Materials Research
National Science Foundation
1800 G Street
Washington, DC 20550

Mr. F. C. Schwenk, RD-T
National Aeronautics and Space
Administration
Washington, DC 20546

M. Zane Thornton
Deputy Director,Institute for
Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

Director
Columbia Radiation Laboratory
Columbia University
538 West 120th Street
New York, NY 10027

Director
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

Director of Laboratories
Division of Applied Sciences
Harvard University
Pierce Hall
Cambridge, MA 02138

Director
Electronics Research Center
The University of Texas
Engineering-Science Bldg. 112
Austin, TX 78712

Director
Electronics Research Laboratory
University of California
Berkeley, CA 94720

Director
Electronics Sciences Laboratory
University of Southern California
Los Angeles, CA 90007

Director
Microwave Research Institute
Polytechnic Institute of New York
333 Jay Street
Brooklyn, NY 11201

Director
Research Laboratory of Electronics
Massachusetts Institute of
Technology
Cambridge, MA 02139

Director
Stanford Electronics Laboratory
Stanford University
Stanford, CA 94305

Stanford Ginzton Laboratory
Stanford University
Stanford, CA 94305

Officer in Charge
Carderock Laboratory
Code 1942.4 Gleissner
David Taylor Naval Ship Research
and Development Center
Bethesda, MD 20084

Dr. Roy F. Potter
3866 Talbot Street
San Diego, CA 92106